UNIVERSITÉ DE STRASBOURG

UFR de mathématique et d'informatique
Université de Strasbourg

UNITÉ DE FORMATION DE RECHERCHE MATHÉMATIQUES ET INFORMATIQUE DE STRASBOURG

MÉMOIRE DE MASTER 2 MATHÉMATIQUES FONDAMENTALES
«EDP et apprentissage»

Contrôle optimal de modèle elliptique, réduction de dimension et apprentissage

---

Amer Jukic

Août 2022

Encadré par le Professeur Emmanuel Franck

Remerciements

En premier lieu je souhaite remercier Monsieur Emmanuel Franck, mon directeur de mémoire, de m'avoir encadré pour ce projet, d'avoir répondu à toutes mes questions, de m'avoir aidé dans l'écriture de ce mémoire ainsi que dans l'écriture du code mais aussi pour ses encouragements qui m'ont permis de mener à bien ce travail.

J'adresse également mes remerciements à ma soeur qui est une source d'inspiration et à mes parents pour leurs encouragements et leur soutien tout au long de mes années d'études.

Je remercie mon père de m'avoir transmis la passion pour les mathématiques et la science. Je remercie ma mère pour tout le réconfort et toute la clarté qu'elle m'a apporté durant mes années d'études.

Par-dessus tout je souhaite les remercier pour leur choix de quitter mon pays d'origine, la Bosnie-Herzégovine, pour un meilleur avenir en France car c'est ce choix qui m'a permis de réaliser mes études.

Je remercie également Zayneb pour son soutien et pour avoir relu plusieurs fois mon mémoire même si le sujet est compliqué à comprendre pour une immunologiste.

Enfin, je remercie les membres du jury de ce mémoire de M2.

Table des matières

Introduction

La compréhension des phénomènes qui nous entourent dans le monde réel ainsi que dans la technologie est basée en grande partie sur l'étude des équations aux dérivées partielles aussi appelées EDP. En effet, grâce à la modélisation de différents phénomènes nous avons pu faire des avancées mathématiques conséquentes et donc obtenir parfois des prévisions physiques avec une grande précision. Les EDP apparaissent en dynamique des structures où sont étudiées par exemple les vibrations c'est-à-dire des oscillations mécaniques autour d'une position, mais aussi en mécanique des fluides ainsi que dans certaines théories de la gravitation, de l'électromagnétique ou même des mathématiques financières. Dans tous ces domaines les équations aux dérivées sont primordiales puisqu'elles modélisent des phénomènes et nous permettent d'avoir des simulations ou des prévisions futures de ces phénomènes. Les modélisations les plus courantes aujourd'hui sont la simulation aéronautique, la synthèse d'images ou la prévision météorologique.

Cependant dans la grande majorité des cas il est impossible de faire quelconque prévision en raison de la nature complexe de l'EDP. Néanmoins grâce à l'apparition d'ordinateurs puissants nous sommes capables d'obtenir des solutions approchées pour ces équations très compliquées en utilisant des méthodes mathématiques très précises.

Ces équations aux dérivées partielles sont donc des équations différentielles dont les solutions sont des fonctions inconnues qui dépendent de plusieurs variables. La particularité d'une EDP est que, sous certaines conditions peu strictes, elle peut posséder plusieurs solutions. C'est pourquoi, dans la grande majorité des cas, on se donne des conditions limites qui nous permettent d'identifier une solution précise et donc de restreindre le domaine des solutions de l'équation.

Il s'agit donc des équations qui sont sous la forme suivante :

$$F(x, u, \frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_d}, ..., \frac{\partial^\alpha u}{\partial x^\alpha}) = 0$$

avec u la fonction inconnue des variables $(x_1, ..., x_d)$ appartenant à un ouvert de $\mathbb{R}^d$, à valeur dans $\mathbb{K}^N$ avec $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$ et $\alpha = (\alpha_1, ..., \alpha_d) \in \mathbb{N}^d$.

Dans cette étude nous allons porter un regard particulier sur les équations aux dérivées partielles elliptiques. Il s'agit des équations de la forme $Lu = f$ où L est un opérateur différentiel linéaire elliptique qui agit le plus souvent sur des fonctions $u \in W^{k,p}(\Omega)$, Ω un ouvert de $\mathbb{R}^N$, de la forme :

$$Lu = \sum_{i,j=1}^{N} a_i j(x) \partial_{i,j}^2 u(x) + \sum_{i=1}^{N} b_i(x) \partial_i u(x) + c(x) u(x)$$

avec $A(x) = (a_{i,j}(x))_{1 \leq i,j \leq N}$ qui est une matrice à coefficients bornés satisfaisant la propriété de coercivité :

$$\exists \lambda > 0 \quad : \quad \sum_{i,j=1}^{N} a_{i,j}(x)\xi \geq \lambda |\xi|^2$$

et f une fonction donnée qu'on appelle le terme source.

Les équations aux dérivées partielles elliptiques les plus typiques sont l'équation de Laplace représentée comme suit :

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0$$

ou bien l'équation qui représente une déformation d'une membrane élastique :

$$-\frac{\partial^2 u}{\partial x^2} + c(x)u(x) = f(x)$$

avec $c(x)$ une caractéristique du matériau constituant le fil et f le chargement auquel le fil est soumis.

Dans cette étude nous allons nous focaliser sur les équations aux dérivées partielles elliptiques et plus particulièrement sur le contrôle optimal, la réduction de dimension pour ces équations et les résolutions par l'apprentissage profond. On détaillera ces notions dans leurs sections respectives.

Pour commencer l'étude nous allons dans un premier temps parler du contrôle optimal de modèle elliptique où nous allons introduire une méthode qui nous permet d'avoir des conditions d'optimalités d'une équation donnée. Ces conditions sont essentielles pour déterminer un contrôle. Nous allons ensuite parler d'un moyen pour calculer le contrôle optimal de manière numérique et nous allons vérifier ce résultat. Dans un deuxième temps allons parler de la réduction de dimension pour les équations aux dérivées partielles où nous allons évoquer rapidement la décomposition en valeurs singulières qui va nous être utile pour parler des méthodes de réduction telles que la méthode POD (proper orthogonal decomposition). On va ensuite montrer une application concrète de la POD et mettre en oeuvre un algorithme qui va nous permettre de trouver la solution optimale d'une équation. Pour finir, nous allons parler de l'apprentissage, en particulier d'une méthode pour approcher les solutions des équations aux dérivées partielles qui vivent en grande dimension.

1 Contrôle optimal de modèle elliptique

La théorie du contrôle analyse les propriétés des systèmes dynamiques sur lesquels on peut agir au moyen d'un contrôle. Le but est d'amener le système d'un état initial donné à un certain état final en respectant des critères du système en question. Autrement dit, il s'agit d'un système dépendant d'un paramètre dynamique appelé le contrôle. L'objectif du contrôle optimal est de déterminer des solutions optimales pour un certain critère d'optimisation. Par conséquent, ce sont des fonctions qui sont généralement soumises à des contraintes.

Contrairement aux équations différentielles ordinaires il n'existe pas de méthodes générales qui nous permettent de résoudre le problème de contrôle d'une équation aux dérivées partielles. En effet, ces dernières sont différentes par leurs natures les unes des autres. Il n'existe donc pas de cadre général fournissant l'existence, l'unicité et la régularité d'une solution. Néanmoins, pour une EDP donnée il est possible d'établir des conditions d'optimalités à la main à travers l'étude de l'état adjoint. Lorsqu'on étudie une équation aux dérivées partielles on prend souvent compte de la fonction coût. Il s'agit d'une fonction qui nous permet de quantifier l'écart entre les prévisions du modèle et l'observation réelle d'un phénomène. Pour calculer le contrôle optimal sur une EDP il faut alors faire de l'optimisation sur la fonction coût. Par conséquent, cela nous permet d'avoir un schéma de calculs pour exprimer le gradient de la fonction coût associée à l'équation aux dérivées partielles qui est essentielle pour l'optimisation.

Dans cette section nous allons commencer par donner quelques rappels sur l'optimisation dans les espaces de Hilbert pour ensuite démontrer un théorème qui va nous permettre de trouver le contrôle optimal d'une EDP elliptique par la méthode de l'adjoint. Nous allons ensuite résoudre le problème du contrôle de cette équation elliptique par un algorithme.

1.1 Optimisation dans les espaces de Hilbert et compléments d'optimisation convexe.

Soit H un espace de Hilbert muni de la norme $||.||$ et du produit scalaire $(.,.)$ et $\mathcal{U} \subset H$ un sous-ensemble convexe fermé.

Dans ce qui suit, la convexité d'une fonctionnelle coût donnée J est essentielle pour étudier le problème du contrôle d'une équation aux dérivées partielles.

Définition 1 *Une fonction $J : H \to \mathbb{R}$ est convexe si*

$$\forall u, v \in H, \forall \theta \in [0,1], J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v)$$

Elle est strictement convexe si

$$\forall u \neq v \in H, \forall \theta \in]0,1[, J(\theta u + (1-\theta)v) < \theta J(u) + (1-\theta)J(v)$$

À partir de la définition nous pouvons énoncer deux propositions qui nous seront utiles pour la suite.

Définition 2 *On dit que J est fortement convexe si et seulement si il existe un $\alpha > 0$ tel que pour tout $u, v \in \mathcal{U}$, $\forall \theta \in [0,1]$*

$$J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v) - \frac{\alpha}{2}\theta(1-\theta)||u-v||^2$$

Proposition 1 *Soit $J : H \to \mathbb{R}$ une fonction différentiable. Alors J est convexe si et seulement si*

$$J(u) \geq J(v) + (\nabla J(v), u - v)$$

Preuve : Soit u, v dans H. La fonction J est convexe si et seulement

$$J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v)$$

Ce qui peut s'écrire $J(\theta u + (1-\theta)v) - J(v) \leq \theta J(u) - \theta J(v)$

Donc

$$\frac{J(v + \theta(u-v) - J(v)}{\theta} \leq J(u) - J(v)$$

En faisant tendre θ vers 0 on a que $J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v)$

si et seulement si $(\nabla J(v), u - v) \leq J(u) - J(v)$

si et seulement si $J(u) \geq J(v) + (\nabla J(v), u - v)$.

Proposition 2 *Soit $J : H \to \mathbb{R}$ une fonction différentiable. Alors les propositions suivantes sont équivalentes :*

1. *J est fortement convexe*
2. *$J - \frac{\alpha}{2}||.||^2$ est convexe*
3. *$\forall u, v \in H$, $(\nabla J(v) - \nabla J(u), v - u) \geq \alpha||v - u||$*

La troisième propriété dit que J est α-elliptique.

Preuve : Montrons l'équivalence 1 et 2. Comme J est fortement convexe on a que $J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v) - \frac{\alpha}{2}\theta(1-\theta)||u-v||^2$.

Par conséquent,

$$J(\theta u+(1-\theta)v)-\frac{\alpha}{2}||\theta u+(1-\theta)v||^2 \leq \theta J(u)+(1-\theta)J(v)-\frac{\alpha}{2}\theta(1-\theta)||u-v||^2-\frac{\alpha}{2}||\theta u+(1-\theta)v||^2$$

Cependant, $\theta J(u) + (1-\theta)J(v) - \frac{\alpha}{2}\theta(1-\theta)||u-v||^2 - \frac{\alpha}{2}||\theta u + (1-\theta)v||^2 = \theta(J(u) - \frac{\alpha}{2}||u||^2) + (1-\theta)(J(v) - \frac{\alpha}{2}||v||^2 - \frac{\alpha}{2}Q(u,v)$

avec $Q(u,v) = \theta(1-\theta)||u-v||^2 + ||\theta u + (1-\theta)v||^2 - \theta||u||^2 - (1-\theta)||v||^2 = 0$

Donc en posant $g(.) = J - \frac{\alpha}{2}||.||^2$ on a que $g(\theta u + (1-\theta)v) \leq \theta g(u) + (1-\theta)g(v)$ d'où la convexité.

Ceci prouve la première équivalence.

La deuxième vient directement du fait que si $g : H \to \mathbb{R}$ est différentiable, alors g est convexe si et seulement si $g(v) \geq g(u) + (\nabla g(u), v-u)$

ou bien si et seulement si $(\nabla g(v) - \nabla g(u), v-u) \geq 0$

D'où la troisième propriété

$$(\nabla J(u) - \nabla J(v), u-v) \ \geq \ \alpha||u-v||^2.$$

Théorème 1 *Soit J continue et fortement convexe, alors le problème $\inf_{\mathcal{U}} J$ possède une unique solution.*

Preuve :

Existence :

Soit $(u_n)_{n\in\mathbb{N}}$ une suite minimisante. On sait que J est fortement convexe, on peut prendre $\theta = 2$ et on a l'existence d'un $\alpha > 0$ tel que pour tout n, m dans $\mathbb{N}$

$$J(\frac{u_n+u_m}{2}) \leq \frac{1}{2}J(u_n) + \frac{1}{2}J(u_m) - \frac{\alpha}{8}||u_n - u_m||^2$$

Cela nous donne :

$$\frac{\alpha}{8}||u_n-u_m||^2 \leq \frac{1}{2}(J(u_n) - J(\frac{u_n+u_m}{2})) + \frac{1}{2}(J(u_m) - J(\frac{u_n+u_m}{2}))$$

Cependant $\mathcal{U}$ est convexe. Il vient alors que $\frac{u_n+u_m}{2} \in \mathcal{U}$.

Par conséquent, $J(\frac{u_n+u_m}{2}) \geq \inf_{\mathcal{U}} J$.

D'où

$$\frac{\alpha}{8}||u_n - u_m||^2 \leq \frac{1}{2}(J(u_n) - \inf_{\mathcal{U}} J) + \frac{1}{2}(J(u_m) - \inf_{\mathcal{U}} J)$$

Or on a que $J(u_n) - \inf_{\mathcal{U}} J \to 0$ quand $n \to \infty$ et $J(u_m) - \inf_{\mathcal{U}} J \to 0$ quand $m \to \infty$ car $(u_n)_n$ est une suite minimisante pour tout n dans $\mathbb{N}$

Donc $u_n - u_m \to 0$ quand $n, m \to +\infty$. Autrement dit la suite $(u_n)_{n\in\mathbb{N}}$ est de Cauchy et donc il existe une limite $u \in \mathcal{U}$ telle que $u_n \to u$ quand $n \to +\infty$. Par la continuité de J il vient que $J(u_n) \to J(u)$ quand $n \to +\infty$. D'où l'existence de la solution.

Unicité :

J étant strictement convexe sur un ensemble convexe il existe alors au plus un minimum global. En effet, si u_1 et u_2 sont deux solutions globales telles que $u_1 \neq u_2$ alors :

$$J(u_1) \leq J(\frac{u_1 + u_2}{2}) \leq \frac{1}{2}J(u_1) + \frac{1}{2}J(u_2) \leq \frac{1}{2}J(u_2) + \frac{1}{2}J(u_2) = J(u_2)$$

Or u_2 est aussi un minimum global. On a donc une contradiction sur $J(u_1) \leq J(u_2)$ et alors $u_1 = u_2$.

Proposition 3 *(Inéquation d'Euler) Si J est convexe et différentiable sur $\mathcal{U}$ qui est convexe, alors x résout $\inf_{\mathcal{U}} J$ si et seulement si $(\nabla J(x), x - y) \geq 0$.*

Preuve : Soit $\epsilon \in]0, 1]$ et $y \in \mathcal{U}$

Pour le sens direct : On se donne un $\epsilon \in]0, 1]$ et $y \in \mathcal{U}$. On a $x + \epsilon(y - x) \in \mathcal{U}$ avec x qui résout $\inf_{\mathcal{U}} J$. Par conséquent on a

$$J(x + \epsilon(y - x)) \geq J(x)$$

En passant le $J(x)$ de l'autre côté et en divisant par ϵ on obtient :

$$\frac{J(x + \epsilon(y - x)) - J(x)}{\epsilon} \geq 0$$

En faisant tendre ϵ vers 0 on a que :

$$(\nabla J(x), y - x) \geq 0$$

Pour le sens inverse :

Si $\frac{J(x+\epsilon(y-x))-J(x)}{\epsilon} \geq 0$ alors pour tout $x \in \mathcal{U}$

$$J(y) \geq J(x) + (\nabla J(x), y - x) \geq J(x)$$

Donc x résout $\underset{\mathcal{U}}{inf} J$.

1.2 Contrôle pour un modèle elliptique

Dans cette section nous allons introduire une équation elliptique et nous allons appliquer la méthode de l'adjoint. Le but est de démontrer qu'il existe une unique solution au problème de contrôle. Pour cela nous avons besoin de calculer l'équation adjointe qui nous permet d'obtenir les conditions d'optimalités de la fonction coût de l'équation aux dérivées partielles grâce au gradient de cette dernière.

Soit Ω un ouvert borné à bord régulier. On désigne par $||.||$ la norme dans $L^2(\Omega)$ et $(.,.)$ le produit scalaire associé à la norme. On désigne par $\mathcal{U}$ le sous-ensemble convexe fermé de $L^2(\Omega)$ comme étant la classe des contrôles admissibles.

On se donne deux fonctions, la fonction source f et z_d la fonction cible. Toutes les deux sont continues de $L^2(\Omega)$. On considère l'équation suivante :

$$\begin{cases} y - \Delta y = f + v & \text{dans} \quad \Omega \\ y = 0 & \text{sur} \quad \partial\Omega \end{cases} \tag{1}$$

La fonctionnelle coût est définie de manière suivante avec $\alpha > 0$:

$$J(v) = \frac{1}{2}||y - z_d||^2 + \frac{\alpha}{2}||v||^2$$

y est la solution de l'équation aux dérivées partielles qu'on veut trouver. z_d est la donnée cible, c'est-à-dire que c'est une observation réelle d'un phénomène physique. α est un coefficient donné et v est le contrôle qu'on chercher à optimiser.

La partie $\frac{1}{2}||y - z_d||^2$ représente une pénalité entre la solution y et la cible z_d mesurée avec la norme de $L^2(\Omega)$. Le deuxième terme $\frac{\alpha}{2}||v||^2$ nous permet de réguler le contrôle v et le coefficient permet d'ajuster cette régularisation dans la fonctionnelle coût.

Notre problème consiste donc à trouver la paire (y, v) qui minimise la fonctionnelle coût sous certaines contraintes.

Autrement dit on cherche à résoudre $\inf_{v \in \mathcal{U}} J(v)$

Il s'agit d'un problème basique de contrôle optimal pour les EDP. Résoudre ce problème de contrôle nécessite l'utilisation des outils issus de l'optimisation. Par conséquent, la résolution numérique de ces problèmes passe aussi par les méthodes d'optimisation. Ces méthodes peuvent être par exemple la méthode de Newton, de gradient projeté à pas fixe ou à pas optimal. Cela dépend en grande partie des détails du problème tels que la structure des contraintes ou l'ensemble des contrôles.

Pour une équation linéaire elliptique l'existence et l'unicité des solutions peut généralement être déduite du résultat abstrait du théorème de Lax-Milgram. Dans notre cas nous ferons la preuve de l'existence et de l'unicité de manière explicite.

Faisons un rappel du théorème de Lax-Milgram et du théorème de régularité elliptique.

Théorème 2 *(Lax-Milgram) Soit a(.,.) une forme bilinéaire continue et elliptique sur un espace de Hilbert H. Soit F une forme linéaire continue sur H. Alors il existe un unique élément u de H tel que*

$$a(u,v) = F(v)\,\forall v \in H$$

De plus si a est symétrique la solution du problème ci-dessus est l'unique solution du problème

$$\inf_{v\in H} J(v) \quad avec \quad J(v) = \frac{1}{2}a(v,v) - F(v)$$

Preuve : Par hypothèse F est une forme linéaire continue sur H. Par le théorème de représentation de Riesz, il existe un unique $G \in H$ tel que pour tout élément v de H on a $F(v) =< G, v >$.

De la même manière par la continuité de a pour $u \in H$ l'application $v \mapsto a(u,v)$ est une forme linéaire continue sur H. Il existe donc un unique $a_u \in H$ tel que $\forall v \in H, \quad a(u,v) =< a_u, v >$.

Écrivons $A : u \in H \mapsto a_u \in H$. Il faut donc montrer qu'il existe un unique $u \in H$ tel que $Au = G$.

Pour cela il suffit de montrer que $A : H \to H$ est bijective.

Premièrement pour u et v dans H et pour un λ réel on a :

$$\forall w \in H, < A(u+\lambda v), w >= a(u+\lambda v, w) = a(u,w)+\lambda a(v,w) =< Au+\lambda Av, w >$$

C'est-à-dire que A est linéaire. De plus, on a la continuité de A. En effet, pour tout u appartenant dans H

$$||Au||^2 =< Au, Au >= a(u, Au) \leq C||u||||A||$$

De plus, puisque a est coercive on à instantanément pour tout u dans H

$$< Au, u >= a(u, u) \geq \alpha||u||^2$$

Il vient alors que $Ker(A) = 0$. De plus $Im(A)$ est fermée. En effet, si on pose $v_n = Au_n$ une suite de $Im(A)$ qui converge vers v dans H on a alors :

$$\alpha||u_n - u_m||^2 \leq a(u_n, u_m) =< A(u_n - u_m), u_n - u_m >\leq ||v_n - v_m||||u_n - u_m||$$

Donc $\alpha||u_n - u_m|| \leq ||v_n - v_m|| \to 0$ lorsque $n, m \to \infty$. En particulier $(u_n)_n$ est de Cauchy dans le Hilbert H donc converge vers u dans H. Avec la continuité de A on a : $Au = lim_{n\to\infty} Au_n = v \in Im(A)$. Donc $Im(A)$ est fermée.

Il nous reste à montrer que A est subjective. Pour cela on vérifie que $(Im(A))^\perp = 0$.

Soit $v \in (Im(A))^\perp$ alors :

$$0 =< Av, v >= a(v, v) \geq \alpha||v||^2$$

donc $v = 0$.

Par conséquent, puisque A est bijective on conclut qu'il existe un unique u dans H tel que pour tout v dans H, $a(u, v) = F(v)$.

Si a est symétrique alors par la coercivité que pour tout v dans H on a

$$J(u + v) = J(u) + a(u, v) - F(v) + \frac{1}{2}a(v, v) = J(u) + \frac{1}{2}a(v, v) \geq J(u) + \frac{\alpha}{2}||v||^2$$

Donc pour $v \neq u$ $J(v) > J(u)$, ce qui dit que u est l'unique minimum de J sur H.

Sous réserve que le domaine Ω et les coefficients de l'équation elliptique soient suffisamment réguliers, la solution existe en un sens plus fort que celui fournit par le théorème de Lax-Milgram.

Théorème 3 *(Régaluarité elliptique)*

Soit $k \in \mathbb{N}$ et Ω un ouvert de classe $C^{k+2}(\Omega)$ avec $a_{i,j} \in C^{k,1}(\overline{\Omega})$ et $c \in C^{k-1,1}(\overline{\Omega})$. Soit $f \in H^k(\Omega)$ et $g \in H^{k+2}(\Omega)$. Soit $u \in H^1(\Omega)$ une fonction telle que

$$Lu = f \text{ au sens de } \mathcal{D}'(\Omega); \quad u - g \in H_0^1(\Omega),$$

avec $Lu = -div(A\nabla u) + cu$ et $A = (a_i, j)$ pour $1 \leq i, j \leq d$

Alors, $u \in H^{k+2}(\Omega)$ et on a l'estimation

$$||u||_{H^{k+2}(\Omega)} \leq C_k(||f||_{H^k(\Omega)} + ||g||_{H^{k+2}(\Omega)})$$

où c_k ne dépend pas de f et de g.

Les théorèmes 2 et 3 nous fournissent l'existence, l'unicité et la régularité d'une équation. Cependant, dans le cadre du contrôle optimal nous voulons minimiser la fonction coût. Pour cela il nous faut l'expression du gradient de cette dernière. C'est tout l'intérêt du théorème qui va suivre, qui nous donne non seulement l'existence et l'unicité de la solution, mais aussi des conditions d'optimalités qui nous sont utiles pour l'étude du contrôle optimal.

A présent nous pouvons montrer le théorème suivant où la méthode de l'adjoint intervient pour résoudre le problème du contrôle pour l'équation elliptique.

Théorème 4 *Le problème du contrôle $inf_{v \in \mathcal{U}} J(v)$ admet une unique solution qui est caractérisée par le système suivant :*

$$\begin{cases} y - \Delta y = f + v & dans \quad \Omega \\ p - \Delta p = y - z_d & dans \quad \Omega \\ y = 0, \quad p = 0 & sur \quad \partial\Omega \\ (p + \alpha v, v - u) \geq 0 & \forall v \in \mathcal{U} \end{cases} \tag{2}$$

Preuve : On commence par montrer que la fonctionnelle J est différentiable sur $L^2(\Omega)$.

Puisque $\partial\Omega$ est C^2 par régularité elliptique avec $k = 0$ f et v sont dans $L^2(\Omega)$ alors l'application $\begin{array}{ccc} \mathcal{U} & \to & H_0^1(\Omega) \cap H^2(\Omega) \\ v & \mapsto & y(x) \end{array}$ est un isomorphisme différentiable.

Par ailleurs, le terme $\frac{\alpha}{2} \int_\Omega v^2 dx$ est différentiable car c'est un terme quadratique en v qui est une fonction de x . Donc la fonctionnelle J est différentiable.

Soit

$$J(v) = \frac{1}{2}\int_\Omega (y(x) - z_d(x))^2 dx + \frac{\alpha}{2}\int_\Omega v^2 dx$$

Pour simplifier les écritures on va écrire : $y(x) = y_v$.

On pose pour tout v de $L^2(\Omega)$:

$$J_1(v) = \frac{1}{2}\int_\Omega (y_v - z_d(x))^2 dx \quad et \quad J_2(v) = \frac{1}{2}\int_\Omega v^2 dx$$

Calculons maintenant la différentielle. On se donne $\epsilon > 0$ et $h \in L^2(\Omega)$.

On a

$$\begin{aligned}\frac{J_1(v+\epsilon h) - J_1(v)}{\epsilon} &= \frac{1}{2\epsilon}\int_\Omega (y_{v+\epsilon h} - z_d)^2 - (y_v - z_d)^2 dx \\ &= \frac{1}{2\epsilon}\int_\Omega (y_{v+\epsilon h} - y_v)(y_{v+\epsilon h} + y_v - 2z_d) dx\end{aligned}$$

Posons $z_h = \frac{y_{v+\epsilon h} - y_v}{\epsilon}$

On a alors :

$$\frac{J_1(v+\epsilon h) - J_1(v)}{\epsilon} = \frac{1}{2}\int_\Omega z_h(y_{v+\epsilon h} + y_v - 2z_d) dx$$

Cependant, $y_{v+\epsilon h}$ résout le même système que $y_v + \epsilon h$, donc par unicité des solutions dans $H^1(\Omega)$ on a que $y_{v+\epsilon h} = y_v + \epsilon h$.

Par conséquent,

$$\frac{J_1(v+\epsilon h) - J_1(v)}{\epsilon} = \frac{1}{2}\int_\Omega z_h(y_v + \epsilon h + y_v - 2z_d) = \int_\Omega z_h(y_v - z_d + \frac{\epsilon\, h}{2}) dx$$

En faisant tendre $\epsilon \to 0$ on obtient que

$$\lim_{\epsilon\to 0} \frac{J_1(v+\epsilon h) - J_1(v)}{\epsilon} = \int_\Omega z_h(y_v - z_d) dx$$

De la même manière on a que :

$$\lim_{\epsilon\to 0} \frac{J_2(v+\epsilon h) - J_2(v)}{\epsilon} = \int_\Omega vh dx$$

Donc :

$$\lim_{\epsilon\to 0} \frac{J(v+\epsilon h) - J(v)}{\epsilon} = \int_\Omega z_h(y_v - z_d) + \alpha\int_\Omega vh dx$$

On remarque que z_h résout l'équation :

$$\begin{cases} z_h - \Delta z_h = \dfrac{f + (v + \epsilon h) - f - v}{\epsilon} = h & \text{dans} \quad \Omega \\ z_h = 0 & \text{sur} \quad \partial\Omega \end{cases} \tag{3}$$

On a alors pour $\delta = y_{v+h} - y_v - z_h$, on a :

$$\begin{cases} \delta - \Delta\delta = f + (v + h) - f - v - h = 0 & \text{dans} \quad \Omega \\ \delta = 0 & \text{sur} \quad \partial\Omega \end{cases} \tag{4}$$

Donc par unicité, $\delta(.) = 0$, c'est-à-dire : $y_{v+h} = y_v + z_h + o(||h||_{L^2(\Omega)})$.

Montrons maintenant que l'application $\begin{matrix} L^2(\Omega) \to H^1(\Omega) \\ h \mapsto z_h \end{matrix}$ est continue.

On multiplie l'équation (2) par z_h et on applique la formule de Green :

$$\int_\Omega z_h^2 dx - \int_\Omega \Delta z_h \; z_h dx = \int_\Omega h \; z_h dx$$

Ce qui nous donne :

$$\int_\Omega z_h^2 dx + \int_\Omega \nabla z_h^2 dx = \int_\Omega h \; z_h dx$$

Par Cauchy-Schwarz on a d'une part :

$$||z_h||^2_{H^1(\Omega)} dx + \int_\Omega |\nabla z_h|^2 dx \leq ||h||_{L^2(\Omega)} ||z_h||_{H^1(\Omega)}$$

Or d'après l'inégalité de Poincaré on a l'existence de $c > 0$ telle que :

$$c||z_h||^2_{H^1} \leq ||\nabla z_h||^2_{L^2}$$

Il vient alors que $min(1, c)||z_h||^2_{H^1} \leq ||h||_{L^2(\Omega)} ||z_h||_{H^1(\Omega)}$

Par conséquent, z_h est continue et c'est la différentielle de y_v.

Calculons maintenant le gradient de J. On a trouvé avant que

$$DJ(v).h = \lim_{\epsilon \to 0} \frac{J(v + \epsilon h) - J(v)}{\epsilon} = \int_\Omega z_h (y_v - z_d) dx + \int_\Omega v h dx$$

Cependant, le terme $\int_\Omega z_h(y_v - z_d)$ est un mauvais terme puisqu'il n'est pas exprimé en fonction de h. On cherche donc la différentielle par la méthode de l'adjoint.

L'équation adjointe est exprimée sous la forme suivante :

$$\begin{cases} p - \Delta p = G \quad \text{dans} \quad \Omega \\ \text{Conditions au bord que l'on va déterminer plus tard} \end{cases} \tag{5}$$

On multiplie l'équation (2) par p et on fait l'intégration par parties :

$$\int_\Omega z_h \; p \; dx + \int_\Omega \nabla z_h \cdot \nabla p \; dx = \int_\Omega h \; p \; dx$$

On multiplie l'équation (3) par z_h et on fait l'intégration par parties :

$$\int_\Omega p \; z_h \; dx + \int_\Omega \nabla p \cdot \nabla z_h \; dx - \int_{\partial\Omega} \frac{\partial p}{\partial n} \; z_h \; d\sigma = \int_\Omega G \; z_h \; dx$$

On choisit $p = 0$ sur $\partial\Omega$ et on obtient alors :

$$\int_\Omega G \; z_h \; dx = \int_\Omega h \, p \, dx$$

Or, on voulait calculer $\int_\Omega z_h \; (y_v - z_d) \; dx$, on obtient donc que $G = y_v - z_d$. Par conséquent l'équation (3) devient :

$$\begin{cases} p - \Delta p = y_v - z_d \text{dans} \quad \Omega \\ p = 0 \text{sur} \partial\Omega \end{cases} \tag{6}$$

Donc

$$DJ(v).h = \int_\Omega h \; p + \alpha \; v \; h \, dx$$

Autrement dit, $\nabla J(v) = p + \alpha v$.

Remarquons que $(\nabla J(v) - \nabla J(u), v - u)_{L^2(\Omega)} \geq \alpha ||v - u||^2_{L^2(\Omega)}$. C'est-à-dire que J est fortement convexe, différentiable et $\mathcal{U}$ est convexe fermé. Donc le problème $inf_{v \in \mathcal{U}}$ possède une unique solution d'après le théorème 1. Cette solution est caractérisée par l'inéquation d'Euler :

$$\forall u \in \mathcal{U} \quad (\nabla J(v), v - u)_{L^2(\Omega)} = (p + \alpha v, v - u)_{L^2(\Omega)} \geq 0$$

Ce qui conclut le théorème.

Ce théorème nous montre qu'on peut écrire des conditions d'optimalités grâce à l'équation adjointe que nous avons introduite. Cela nous permet d'avoir l'expression du gradient de la fonctionnelle coût :

$$\nabla J(v) = p + \alpha v$$

Et par conséquent d'avoir un unique contrôle optimal. Il est possible de généraliser cette méthode à des formulations variationnelles plus générales.

On parle d'état adjoint car lorsqu'on pose notre équation avec p on fait intervenir l'adjoint, au sens des distributions de l'opérateur qu'on considère. Bien évidemment lorsque l'opérateur est auto-adjoint il n'est pas nécessaire de faire toute la démarche précédente car l'équation adjointe en p est la même que l'équation d'état en y.

1.3 Partie numérique du contrôle optimal

Par la suite nous allons construire un algorithme qui va nous permettre de résoudre l'équation (2) de façon numérique. L'objectif est d'avoir un code qui nous renvoie explicitement la solution de notre contrôle optimal v.

À partir du théorème 2 nous pouvons créer un algorithme de minimisation basé sur la méthode de gradient à pas constant qui nous retourne le contrôle optimal v. Avant cela, nous allons utiliser la méthode des différences finies qui va nous permettre de résoudre de manière numérique l'équation d'état y et l'équation adjointe p. Commençons alors cette section avec quelques rappels sur les différences finies.

1.3.1 Différences finies pour des problèmes de type elliptique en dimension 1

L'objectif de la méthode des différences finies est de calculer une approximation numérique des valeurs des dérivées d'une fonction.

Soit $L > 0$. En général on se place dans $[0, L]$ que l'on discrétise à l'aide de la grille définie par ses sommets :

$$x_n = nh \quad avec \quad n = 0, ..., N \; et \; (N+1)h = L$$

Avec h le pas de la grille.

On se place en $0 < x_n < L$ avec $1 < n < N$ et on pose $h = \frac{L}{N+1}$. On veut approcher l'équation (2) par différences finies. Pour cela, on discrétise l'équation (2) ainsi que les conditions limites. En notant $y(x_n) = y_n$ et $f(x_n) = f_n$ on obtient dans notre cas :

$$\begin{cases} y_n - \frac{y_{n-1}-2y_n+y_{n+1}}{h^2} = f_n & \text{dans} \quad \Omega \\ y_n = 0 & \text{sur} \quad \partial\Omega \end{cases} \tag{7}$$

Dans notre situation nous allons utiliser la formulation vectorielle des différences finies.

On a alors un schéma qui est :

$$\begin{cases} y_0 = 0 \\ y_1 - \frac{y_0-2y_1+y_2}{h^2} = f_1 \\ y_2 - \frac{y_1-2y_2+y_3}{h^2} = f_2 \\ y_3 - \frac{y_2-2y_3+y_4}{h^2} = f_3 \\ .... \\ y_N - \frac{y_{N-1}-2y_N+y_{N+1}}{h^2} = f_N \\ y_{N+1} = 0 \end{cases} \tag{8}$$

En éliminant la première et la dernière équation substituées dans la deuxième et avant-dernière équation on obtient alors le système de N inconnue

$$A\mathbf{Y} = \mathbf{F}$$

$$\text{avec } \mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, \mathbf{F} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix} \text{ et } A = \frac{1}{h^2} \begin{bmatrix} 1 & -1 & 0 & 0 & ... & 0 \\ -1 & 2 & 1 & 0 & ... & 0 \\ 0 & -1 & 2 & -1 & 0 & ... \\ ... & ... & ... & ... & ... & \\ 0 & ... & ... & 0 & -1 & 1 \end{bmatrix} + I_N$$

1.3.2 Algorithme et résultats

On construit l'algorithme qui calcule le contrôle optimal de manière suivante :

Algorithm 1 Trouver v optimal de $y - \Delta y = f + v$

Require: $v^0 \in \mathcal{U}, \epsilon > 0, \rho > 0, Res = 1$

while $Res \geq \epsilon$: **do**
  $y^k - \Delta y^k \leftarrow f + v^k$
  $p^k - \Delta p^k \leftarrow y^k - z_d$
  $v^{k+1} \leftarrow v^k - \rho(p^k + \alpha v^k)$
  $Res \leftarrow ||v^{k+1} - v^k||_2$
end while
return v^{k+1}

Il s'agit d'une méthode de gradient à pas constant. A chaque itération on calcule y^k, p^k et on met à jour v^k. Cela nous permet de calculer le gradient à chaque itération avec la solution de l'adjoint p^k et le contrôle v^k. Autrement dit, à chaque itération on calcule $\nabla J(v^k) = p^k + \alpha v^k$. Avec le gradient on peut alors actualiser la direction de la descente pour la méthode et donc se rapprocher d'une meilleure façon du contrôle optimal.

On se donne en entrée une fonction v^0, un pas $\rho > 0$, une tolérance $\epsilon > 0$ et un résidu $res = 1$. Pour résoudre $y^k - \Delta y^k = f + v^k$ et $p^k - \Delta p^k = y^k - z_d$ dans la boucle on utilise les différences finies. On calcule alors la nouvelle valeur du contrôle v^{k+1} tout en vérifiant que le résidu est plus grand que la tolérance.

On décide de prendre $N = 1000$.

Pour tester notre algorithme 1 on va prendre comme source $f(x) = sin(2\pi x)$ avec $x \in [0, 1]$, $z_d(x) = sin(2\pi x)$. On choisit une tolérance $\epsilon = 10^{-5}$, $\alpha = 10^{-5}$ et $\rho = 10$.

Premièrement on vérifie que notre méthode des différences finies fonctionne. Avec le terme source $f(x) = sin(2\pi x)$ on s'attend à avoir la solution théorique

$$y(x) = \frac{1}{1 + 4\pi^2} sin(2\pi x)$$

.

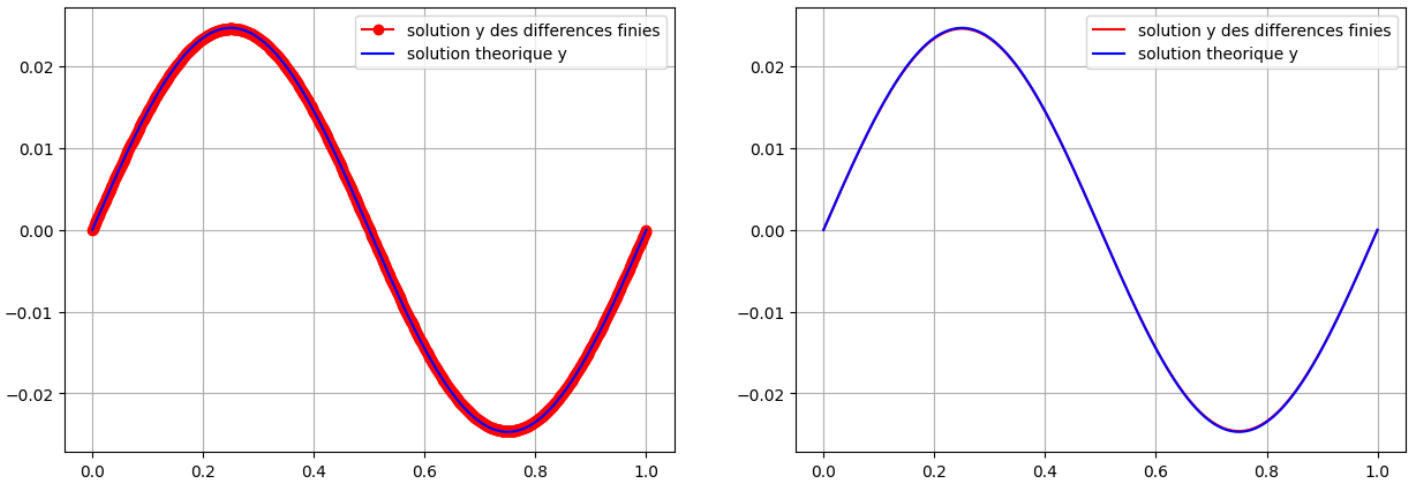

FIGURE 1 – Solution théorique et la solution numérique qui est fournie par la méthode des différences finies

```
Erreur en norme L2 entre la solution théorique et numérique
0.002148115962847123
```

On observe que la solution théorique est bien approchée par notre méthode des différences finies. Nous pouvons maintenant exécuter l'algorithme 1 pour chercher le contrôle optimal de notre équation (2).

On décide de donner en entrée $v^0(x) = 3\pi^2 sin(2\pi x) + \gamma sin(2\beta\pi x)$ avec $\gamma = 1$ et $\beta = 1$.

Le contrôle optimal que l'algorithme doit nous fournir doit être égal à $v(x) = 4\pi^2 sin(2\pi x)$.

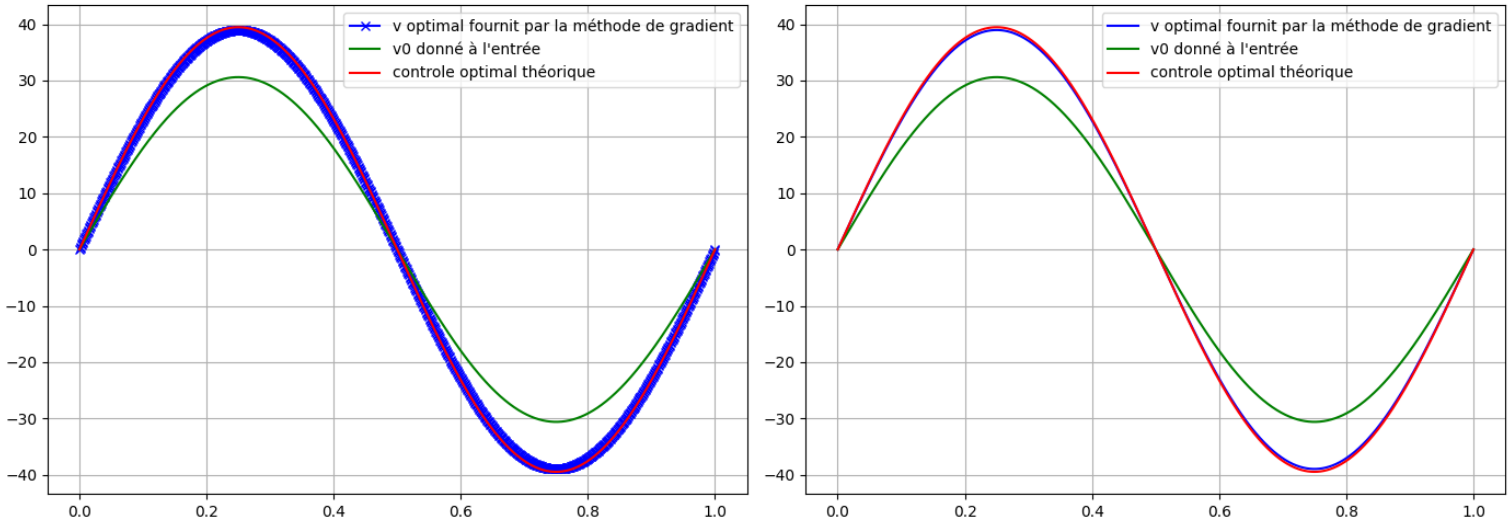

FIGURE 2 – Contrôle optimal numérique calculé par la méthode de gradient et contrôle optimal théorique

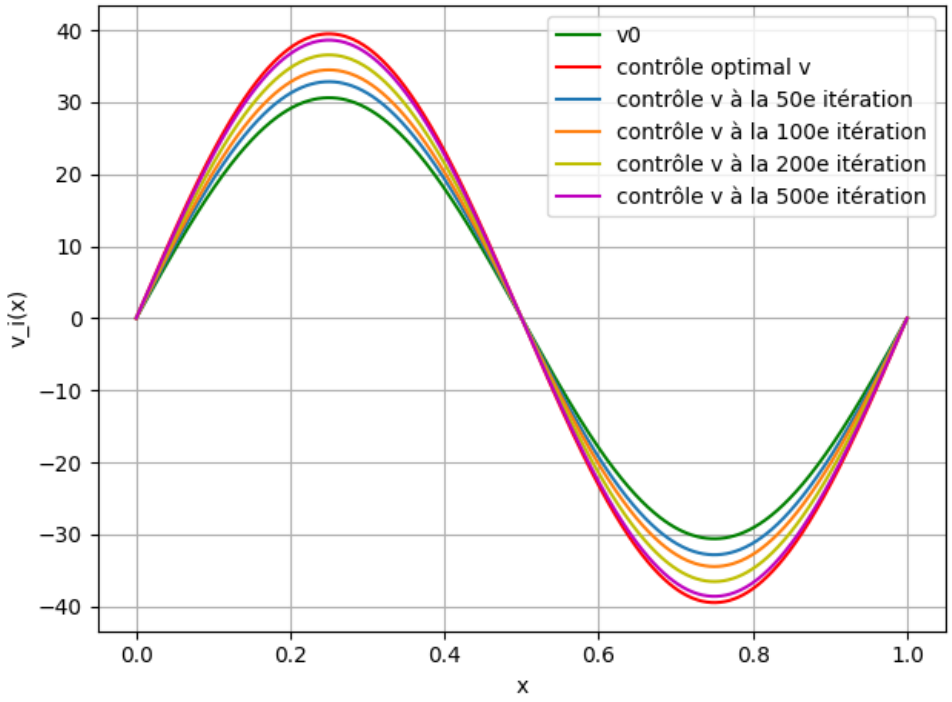

FIGURE 3 – Affichage des contrôles v à chaque étape d'itération de la méthode de gradient

La figure 2 nous montre que le contrôle optimal est bien approché par la méthode de gradient décrit par l'algorithme 1. Par ailleurs on observe bien dans la figure 3 que la méthode fait converger les solutions au fil des itérations vers le contrôle optimal. Puisque nous avons choisi $f(x) = sin(2\pi x)$ et $z_d(x) = sin(2\pi x)$ on

s'attend à ce que la fonctionnelle $J(v)$ soit proche de 0 au fil des itérations. En effet, on a choisi $\alpha = 10^{-5}$ et le contrôle optimal est $v(x) = 4\pi^2 sin(2\pi x)$. Il vient alors que $y(x) = sin(2\pi x)$, c'est-à-dire que $||y - z_d||^2 = 0$. Donc il reste que $J(v) = \frac{\alpha}{2}||4\pi^2 sin(2\pi x)||^2$.

Or,

$$\int_0^1 sin(2\pi x)^2 dx = [\frac{x}{2} - \frac{sin(4\pi x)}{8\pi}]_0^1 = \frac{1}{2}$$

Par conséquent,

$$J(v) = \frac{\alpha}{2}\, 16\, \pi^4\, \frac{1}{2} \approx 0.0038$$

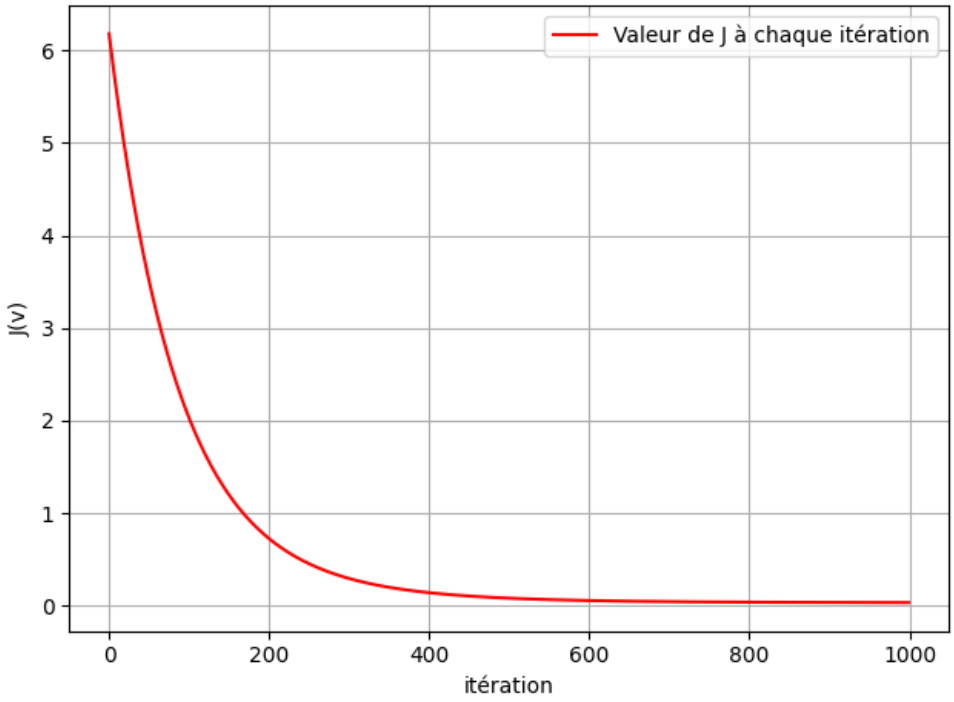

FIGURE 4 – Valeur de $J(v)$ à chaque étape d'itération

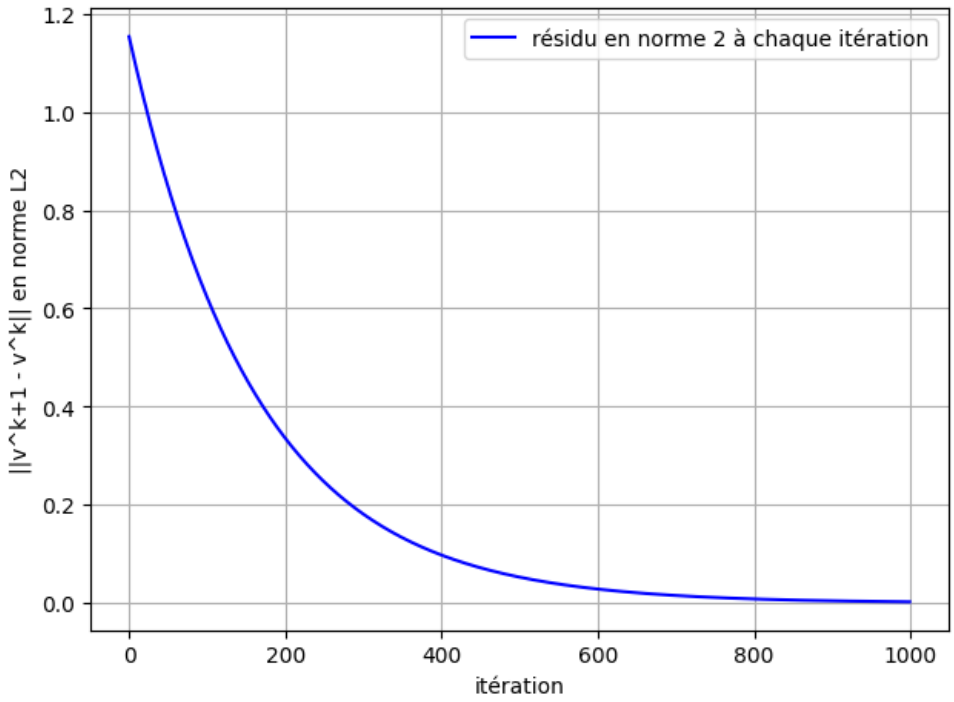

FIGURE 5 – Valeur du résidu à chaque étape d'itération

On peut voir aux figures 3,4 et 5 que la méthode converge bien vers le contrôle optimal.

C'est donc à travers l'étude du gradient de la fonction coût que nous pouvons approcher numériquement le contrôle optimal. C'est pourquoi introduire l'équation adjointe est nécessaire. En effet, cela nous donne des conditions d'optimalités qui sont essentielles à l'étude théorique de la solution mais aussi un moyen de calculer le gradient explicitement et donc pour notre algorithme de pouvoir mettre à jour la direction de descente.

2 Réduction de dimension

Dans cette partie nous allons résoudre le problème de minimisation $\min_v J(v)$ avec $J(v) = \frac{1}{2}||y - z_d||^2 + \frac{\alpha}{2}||v||^2$ en faisant une réduction de dimension.

En vertu du théorème de Lax-Milgram pour notre équation aux dérivées partielles elliptique (1) nous pouvons écrire grâce à la formulation variationnelle $a(y, \varphi) = F(\varphi)$ avec les relations suivantes :

$$a(y,\varphi) = \int_\Omega y\varphi dx + \int_\Omega \nabla y \cdot \nabla \varphi dx \quad et \quad F(\varphi) = \int_\Omega (f+v)\varphi dx$$

Les méthodes de réduction de dimensions pour les équations aux dérivées partielles sont des techniques qui permettent de réduire les coûts des calculs liés à la résolution numérique de problèmes vivant dans une grande dimension. L'intérêt principal est d'être capable de résoudre une même équation mais en dimension beaucoup plus petite que celle de départ et de garder les caractéristiques importantes de l'équation en question. Par exemple, si on a une équation avec une fonction coût sous contrainte, la contrainte doit être vérifiée en petite et grande dimension. Il faut être capable d'appliquer une compression sur l'équation et de projeter toutes les informations en basse dimension pour travailler dans le domaine réduit. Ceci nous permet de gagner un temps de calcul considérable quant à la résolution de l'équation. Une fois la solution trouvée, il faut ensuite la décompresser, c'est-à-dire la projeter en grande dimension. Dans cette grande dimension la solution doit vérifier ensuite les conditions de bords, les conditions limites etc.

Dans le cas de la méthode de gradient de l'algorithme 1 on peut voir qu'on résout beaucoup d'équations aux dérivées partielles duales et primales. Cela utilise beaucoup de ressources en matière de coût de calcul, notamment si la dimension est très grande. Le principe des méthodes de réduction est d'extraire les données

les plus pertinentes afin d'alléger ces coûts de calculs tout en ayant une bonne approximation de la solution.

L'objectif est donc de construire un modèle réduit où les résultats obtenus dans la dimension initiale restent valides. Pour commencer, nous allons étudier le cas dans un contexte plus général. On considère donc le modèle $a(y, \varphi; \mu) = F(\varphi; \mu)$ avec μ qui est un vecteur de paramètres issus du domaine Ω.

On peut alors faire l'hypothèse qu'il existe un sous-espace vectoriel $vect(\phi_i) \subset \mathbb{R}^n$ pour $i = 1, ..., m$ avec $m << n$ où n est la dimension de notre problème en grande dimension. Par conséquent, on peut représenter les solutions en grande dimension par des données vivant en petite dimension. On peut écrire ceci de la manière suivante $a(y, \varphi; \mu) = a(y_m, \varphi; \mu)$, c'est-à-dire qu'on veut avoir $y = \Phi y_m$ où Φ est l'opérateur qui relie la petite dimension à la grande dimension et y_m la solution en petite dimension.

2.1 Décomposition en valeurs singulières (SVD)

Avant de parler des méthodes de réduction nous allons introduire quelques notions à propos de la décomposition en valeurs singulières qui nous seront utiles dans les preuves plus tard. En effet, la SVD d'une matrice nous donne des informations utiles telles que le rang, la norme ou bien les espaces nulles. Le plus important dans la SVD est l'optimalité c'est-à-dire la méthode qui permet d'obtenir la meilleure approximation de rang faible d'une matrice au sens de la norme euclidienne. Puisque cette décomposition existe pour toutes les matrices, qu'elles soient carrées, rectangulaires, singulières, nous pouvons utiliser les propriétés de cette dernière pour obtenir une base POD.

Proposition 4 *Pour toute matrice $X \in M_{m\times n}(\mathbb{R})$ il existe des matrices orthogonales $U \in M_{mxm}(\mathbb{R})$, $V \in M_{n\times n}(\mathbb{R})$ ainsi qu'une matrice diagonale*

$$\Sigma = diag(\sigma_1, ..., \sigma_p) \in M_{m\times n}(\mathbb{R}), \quad p = min(m, n)$$

avec $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$

telles que

$$X = U\Sigma V^T$$

Ici on note par les $\sigma_i, \quad i = 1, ..., p$ les valeurs singulières de X

Proposition 5 *Soit $X = U\Sigma V^T$ la décomposition en valeurs singulières de $X \in \mathbb{R}^{m\times n}$ avec les valeurs singulières $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_d \geq \sigma_{d+1} = ... = \sigma_n = 0$.*

Pour $k \leq d = Rang(X)$ on pose :

$$X_k = U\Sigma_k V^T$$

où $\Sigma_k = diag(\sigma_1, ..., \sigma_k, 0, ..., 0) \in \mathbb{R}^{m\times n}$. On a alors :

1. $Rang(X_k) = k$

2. $||X - X_k||_2 = \sigma_{k+1}$

3. $||X - X_k||_2 = \min\limits_{Rang(B)\leq k} ||X - B||_2$

La troisième propriété nous dit que X_k est la meilleure approximation de X de rang $\leq k$.

2.2 Méthode POD (Proper orthogonal decomposition)

Cette méthode consiste à chercher des modes capables de donner la meilleure représentation possible aux simulations qu'on se donne. On considère une matrice de snapshots qui contient un nombre d'exemples issus de notre simulation. On note cette matrice par X de taille $n \times d$. L'objectif principal est de faire une compression puis une décompression des données. Pour cela, on doit construire un opérateur Φ_K qui nous donnera le décodeur (Φ_K) et l'encodeur (Φ_K^T). Ceci nous permettra d'exprimer $\Phi_K\Phi_K^T X$ qui traduit une compression faite par l'encodeur puis une décompression faite par le décodeur.

Définition 3 *On considère la matrice de snapshots $X \in \mathcal{M}_{n\times d}$. La méthode POD construit l'opérateur Φ_K qui est solution du problème :*

$$\min_{\Phi_K \in \mathbb{R}^{n\times K}} ||X - \Phi_K\Phi_K^T X||_F^2$$

avec la contrainte $\Phi_K^T\Phi_K = I_d$ et $||.||_F$ la norme de Frobenius.

Ce problème revient à résoudre le problème de minimisation de l'erreur de reconstruction après la compression et décompression. Cela nous mène au Lemme suivant :

Lemme 1 *La solution de l'opérateur $\Phi_K \in \mathcal{M}_{n\times K}$ est donnée par U_K qui sont les K premiers vecteurs de la matrice U issue de la SVD de $X = U\Sigma V^T$.*

Preuve : On considère la matrice de snapshots X. On fait la décomposition en valeurs singulières

$$X = U\Sigma V^T$$

On rappelle que Σ est la matrice des valeurs propres de XX^T et X^TX, la matrice V la matrice des vecteurs propres de X^TX et U la matrice des vecteurs propres de XX^T. Puisque les XX^T sont symétriques réelles, ses vecteurs propres sont orthogonaux. Par le théorème de la SVD la décomposition reste valide pour une restriction aux K premiers éléments qui est la meilleure approximation possible de rang K de la matrice X.

$$X_K = U_K\Sigma_K V_K$$

Par conséquent on a :

$$\underset{rg(B)=K}{min} ||X - B||_F^2 = ||X - X_K||_F^2$$

Si on prend $\Phi_K = U_K$ on a que grâce à l'orthogonalité :

$$\Phi_K\Phi_K^T X_K = U_K U_K^T X_K = U_K U_K^T U_K \Sigma_K V_K = X_K$$

Ce qui montre que U_K est la meilleure approximation de rang K de X. Par conséquent U_K est la solution du problème de minimisation.

2.2.1 Projection de Galerkin

Une fois que l'encodeur et le décodeur sont déterminés, il nous faut construire le modèle sur un espace réduit. C'est là qu'intervient la méthode de projection de Galerkin.

On introduit le résidu : $r(y, \varphi; \mu) = a(y, \varphi; \mu) - F(\varphi; \mu)$ et on veut que

$r(y, \varphi; \mu) = 0$.

A partir de là on peut écrire le résidu réduit $\hat{r}(y_m, \varphi; \mu) = r(\Phi_K y_m, \varphi; \mu)$

Ceci nous permet d'écrire le résultat suivant :

Définition 4 *La projection de Galerkin est donnée par*

$$\dot{y} = \underset{\eta \in vect(\phi_i)}{argmin} ||r(\eta, \varphi; \mu)||_2^2$$

qui est équivalent à :

$$\dot{y_m} = \underset{\eta_m \in \mathbb{R}^m}{argmin} ||\hat{r}(\eta_m, \varphi; \mu)||_2^2$$

Cela revient à trouver l'expression de notre dérivée sur les variables réduites. Le but est alors de minimiser le résidu réduit. Un résidu nul signifie que l'approximation ne fait aucune erreur une fois incorporée dans l'équation. En pratique ce n'est pas possible mais l'idée naturelle est d'essayer de minimiser le résidu issu de notre approximation introduite dans l'équation aux dérivées partielles.

2.2.2 Application de la réduction à notre équation

On veut maintenant construire le modèle sur les données réduites de notre équation (1). Il s'agit d'un cas très simple. L'intérêt est de montrer l'efficacité de la méthode.

On suppose que l'on a construit notre décodeur Φ. En utilisant le résidu sur notre équation on obtient $r(y, f) = y - \Delta y - f - v$.

On remplace y par son approximation affine on écrit $y = \sum_{i=1}^{K} \alpha_i \phi_i = \Phi\alpha$. Par conséquent en basse dimension y est représentée par α, Δy par $A\alpha$ où A est la Laplacien réduit et v par α_v.

Puisque nous avons choisi le terme source comme étant $sin(2\pi x)$ il est possible de construire notre contrôle v comme un élément vivant dans un espace de Fourier. On peut alors choisir des fonctions sinus avec des différentes fréquences et des coefficients qui varient entre -1 et 1 afin d'avoir une large gamme de choix pour construire notre contrôle optimal.

On veut donc notre v écrit de la façon $v = a_1\psi_1 + a_2\psi_2 + a_3\psi_3 + a_4\psi_4 + a_5\psi_5$ avec $\psi_1 = sin(2\pi x)$, $\psi_2 = sin(4\pi x)$, $\psi_3 = sin(6\pi x)$, $\psi_4 = sin(8\pi x)$,$\psi_5 = sin(10\pi x)$ et les $a_i \in [-1, 1]$.

Il nous faut donner l'expression de f,z_d et de v dans la base réduite $\mathcal{B} = (\phi_1, ..., \phi_K)$. Pour cela on utilise la projection de Galerkin. On écrit :

$$\Pi_{\mathcal{B}}(f) = \sum_{i=1}^{K} \alpha_f^i \phi_i$$

où $\Pi_{\mathcal{B}}(f)$ est la projection de Galerkin de f sur $\mathcal{B}$.

Le but étant de trouver les $(\alpha_f^1, ..., \alpha_f^K)$ tel que

$$\sum_{i=1}^{K} \alpha_f^i \phi_i = f$$

Pour cela, on multiplie cette relation par les $(\phi_1, ..., \phi_i)$ qui sont orthonormaux. Il vient alors que pour tout $i \in \{1, ..., K\}$

$$\alpha_f^i = \frac{< f, \phi_i >}{||\phi_i||} =< f, \phi_i >$$

On obtient de la même manière les α_{z_d} qui sont la représentation de z_d dans la base réduite $\mathcal{B}$ et v qui s'écrit donc $\alpha_v = \sum_{i=1}^{5} a_i \alpha_{\psi_i}$ avec α_{ψ_i} la représentation de ψ_i en basse dimension.

Par conséquent, grâce à l'identité du résidu on a $\hat{r}(\alpha, \alpha_f) = r(\Phi\alpha, \Phi\alpha_f)$

Ce qui nous donne :

$$\alpha - A\alpha = \alpha_f + \sum_{i=1}^{K} a_i \alpha_{\psi_i} = \alpha_f + \alpha_v$$

Cette relation est tout simplement l'équation d'origine (1) mais projetée dans la base réduite.

Ensuite, résoudre notre problème $min_v \frac{1}{2}||y - z_d||^2 + \frac{\beta}{2}||v||^2$ revient à résoudre :

$$\min_{a_1,a_2,a_3,a_4,a_5} \frac{1}{2}||\alpha - \alpha_{z_d}||_2^2 + \frac{\beta}{2}\sum_{i=1}^{K} |a_i|^2 = \min_{a_1,a_2,a_3,a_4,a_5} \frac{1}{2}||\alpha - \alpha_{z_d}||_2^2 + \frac{\beta}{2}||\alpha_v||^2$$

sous la contrainte $\alpha - A\alpha = \alpha_f + \sum_{i=1}^{K} a_i \alpha_{\psi_i}$.

Pour simplifier les notations, on peut prendre $\beta = 1$.

Pour résoudre ceci on utilise les multiplicateurs de Lagrange et on obtient la fonction :

$$L(a_1, a_2, a_3, \lambda_1, \lambda_1, \lambda_1) = \frac{1}{2}||\alpha - \alpha_{z_d}||_2^2 + \frac{1}{2}\sum_{i=1}^{K} |a_i|^2 + \sum_{i=1}^{K} \lambda_i(\alpha - A\alpha - \alpha_f - a_i\alpha_{\psi_i})$$

En dérivant par rapport aux a_i et λ_i on obtient :

$$\frac{\partial L}{\partial a_i} = a_i - \lambda_i \alpha_{\psi_i} \quad \frac{\partial L}{\partial \lambda_i} = \alpha - A\alpha - \alpha_f - a_i \alpha_{\psi_i}$$

ceci nous mène à au système suivant :

$$\begin{cases} a_i = \lambda_i \alpha_{\psi_i} \\ \alpha - A\alpha - \alpha_f - a_i \alpha_{\psi_i} = 0 \end{cases} \tag{9}$$

ce qui donne :

$$\begin{cases} a_i = \lambda_i \alpha_{\psi_i} \\ \alpha - A\alpha - \alpha_f - \lambda_i \alpha_{\psi_i}^2 = 0 \end{cases} \tag{10}$$

Finalement on obtient :

$$\begin{cases} a_i = (\alpha - A\alpha - \alpha_f)(\alpha_{\psi_i})^{-1} \\ \lambda_i = (\alpha - A\alpha - \alpha_f)(\alpha_{\psi_i})^{-2} \end{cases} \tag{11}$$

On injecte les $a_i \alpha_{\psi_i}$ dans l'équation réduite de J et on fait la minimisation sur α ce qui nous donne :

$$\frac{1}{2}||\alpha - \alpha_{zd}||_2^2 + \frac{1}{2}||(I - A)\alpha - \alpha_f||_2^2$$

On a

$$L(\alpha) = \frac{1}{2}(\alpha^T\alpha - \alpha^T\alpha_{zd} - \alpha_{zd}^T\alpha + \alpha_{zd}^T\alpha_{zd}) + \frac{1}{2}(\alpha^T(I-A)^T(I-A)\alpha - \alpha^T(I-A)^T\alpha_f - \alpha_f^T(I-A)\alpha + \alpha_f^T\alpha_f)$$

En faisant le gradient par rapport à α et en l'annulant on obtient :

$$\alpha - \frac{1}{2}(\alpha_{zd} + \alpha_{zd}^T) + (I - A)^T(I - A)\alpha + \frac{1}{2}((I - A)^T\alpha_f + \alpha_f^T(I - A))$$

Ce qui donne

$$(I + (I - A)^T(I - A))\alpha = \frac{1}{2}(\alpha_{zd} + \alpha_{zd}^T) + \frac{(1}{2}((I - A)^T\alpha_f + \alpha_f^T(I - A)))$$

d'où

$$\alpha = (I + (I - A)^T(I - A))^{-1}(\frac{1}{2}(\alpha_{zd} + \alpha_{zd}^T) + \frac{1}{2}((I - A)^T\alpha_f + \alpha_f^T(I - A)))$$

donc,

$$\alpha = (I + (I - A)^T(I - A))^{-1}(\alpha_{zd} + (I - A)^T\alpha_f)$$

Par conséquent on obtient :

$$\alpha_v = \sum_{i=1}^{K} a_i\psi_i = (I + (I - A)^T(I - A))^{-1}(I - A)(\alpha_{z_d} + (I - A)^T\alpha_f) - \alpha_f$$

En essayant de résoudre $\frac{1}{2}||\alpha - \alpha_{zd}||_2^2 + \frac{1}{2}||(I - A)\alpha - \alpha_f||_2^2$ avec la méthode des moindres carrés on obtient l'équation normale modifiée :

$$(I + (I - A)^T(I - A))\alpha = \alpha_{z_d} + (I - A)^T\alpha_f$$

Et on se retrouve avec la même solution.

Finalement, lorsqu'on prend en compte notre β on obtient les solutions en dimension réduite suivantes :

$$\alpha = (I + \beta(I - A)^T(I - A))^{-1}(\alpha_{zd} + \beta(I - A)^T\alpha_f)$$

$$\alpha_v = \sum_{i=1}^{K} a_i\alpha_{\psi_i} = (I + \beta(I - A)^T(I - A))^{-1}(I - A)(\alpha_{z_d} + \beta(I - A)^T\alpha_f) - \alpha_f$$

2.3 Partie numérique, observation et vérification

Dans cette partie nous allons mettre en oeuvre un code pour vérifier les résultats théoriques obtenus dans la partie précédente. On rappelle rapidement qu'on a choisi $N = 1000$ étant la dimension et $h = \frac{1}{N+1}$. L'idée est d'avoir une matrice de snapshots qui représente un espace de Fourier pour ensuite être capable de construire notre solution via une méthode POD. Pour cela nous allons procéder de la manière suivante : on va tirer aléatoirement des e_i dans $[-1, 1]$ pour $i = 1, .., 5$ et on va poser

$$s = \sum_{i=0}^{5} e_i \, sin(2\pi \, \omega_i \, x) = \sum_{i=0}^{5} ei \, \psi_i$$

avec $\omega_i = 1, 2, 3, 4, 5$. Ceci va nous permettre d'avoir des vecteurs S_j pour $j = \{1, ..., 500\}$ avec lesquels on va résoudre l'équation :

$$u_j - \Delta u_j = S_j$$

Ainsi on aura notre matrice de snapshots $M_s = [S_1, .., S_j, .., u_1, .., u_j, ...]$ de taille 1000x1000.

Algorithm 2 Construction de la matrice M_s

Require: $\psi_1, \psi_2, \psi_3, \psi_4, \psi_5,$

for $j = 1, ..., 500$ **do**
  $res = 0$
  $e_j = random.uniform(-1, 1, 5)$
  for $i = 1, 5$ **do**
    $res = res + ei\,\psi_i$
  end for
  $S_j = res$

  Solve $u_j - \Delta u_j = S_j$
  $M_s = stack(s_j, u_j)$
end for
return M_s

Algorithm 3 Méthode POD

Require: Matrice de snapshots M

On applique la SVD à la matrice M
$M = U\Sigma V^T$
On choisit les K premieres colonnes de U comme les modes POD :
$\Phi_K = U(:, 1 : K)$
return Φ_K

A partir de cette matrice nous allons construire Φ par la méthode de POD détaillée ci-dessus et nous pourrons projeter l'équation $y - \Delta y = f + v$ dans la base de Φ ce qui nous donnera notre équation réduite $\alpha - A\alpha = \alpha_f + \alpha_v$.

Cela va nous permettre de calculer numériquement les solutions théoriques que nous avons obtenues précédemment.

Premièrement, nous allons récupérer le contrôle optimal v que nous avons trouvé avec la méthode de gradient et nous allons appliquer la compression avec Φ pour vérifier que le résultat théorique correspond au contrôle optimal compressé. On vérifie donc qu'on a bien

$$\alpha_v = (I + \beta(I - A)^T(I - A))^{-1}(I - A)(\alpha_{z_d} + \beta(I - A)^T\alpha_f) - \alpha_f$$

Ensuite nous allons décompresser la solution théorique pour étudier la correspondance avec le v optimal en grande dimension.

Faisons une observation de notre code.

Lorsqu'on trace les valeurs propres de notre méthode de POD on observe qu'on atteint 0 au bout de 2 modes.

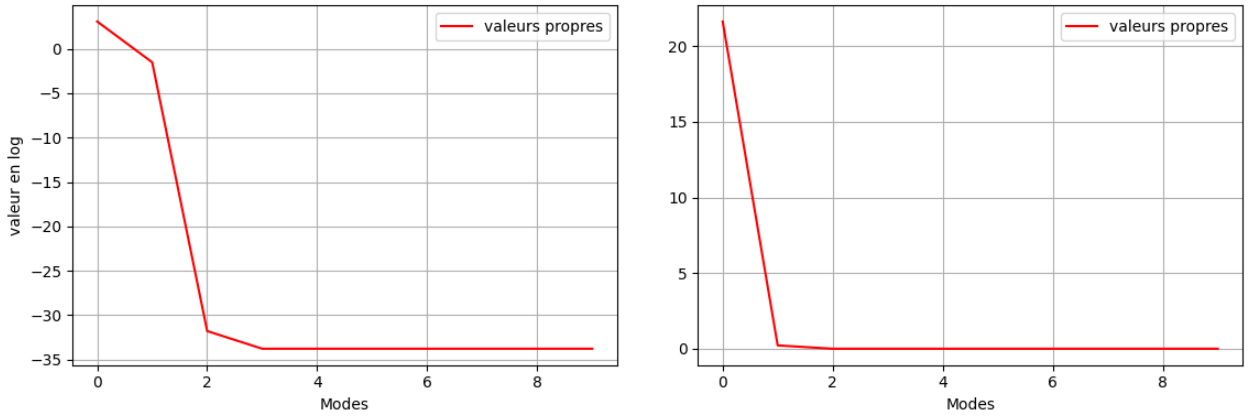

FIGURE 6 – Affichage de la décroissance des valeurs propres

On choisit alors 2 modes pour construire notre Φ et on observe que c'est bien une matrice orthogonale :

```
On vérifie que phi est une matrice orthogonale :

[[1.00000000e+00 2.77555756e-17]
 [2.77555756e-17 1.00000000e+00]]
```

Avec ceci on fait d'une part la compression de notre contrôle optimal v et d'autre part on calcule numériquement notre α_v théorique. On étudie ensuite la contrainte $\alpha - A\alpha - \alpha_f - \alpha_v = 0$ pour savoir si elle est vérifiée :

```
La contrainte avec le résultat théorique alpha_v:
[0.00000000e+00 2.27373675e-13]

La contrainte de v optimal compressé :
[ 0.00000000e+00 -2.27373675e-13]
```

On remarque que la contrainte vaut 0 ce qui nous laisse penser que notre compression du contrôle optimal v ainsi que le calcul numérique de α_v sont justes. Observons les graphes de ces derniers.

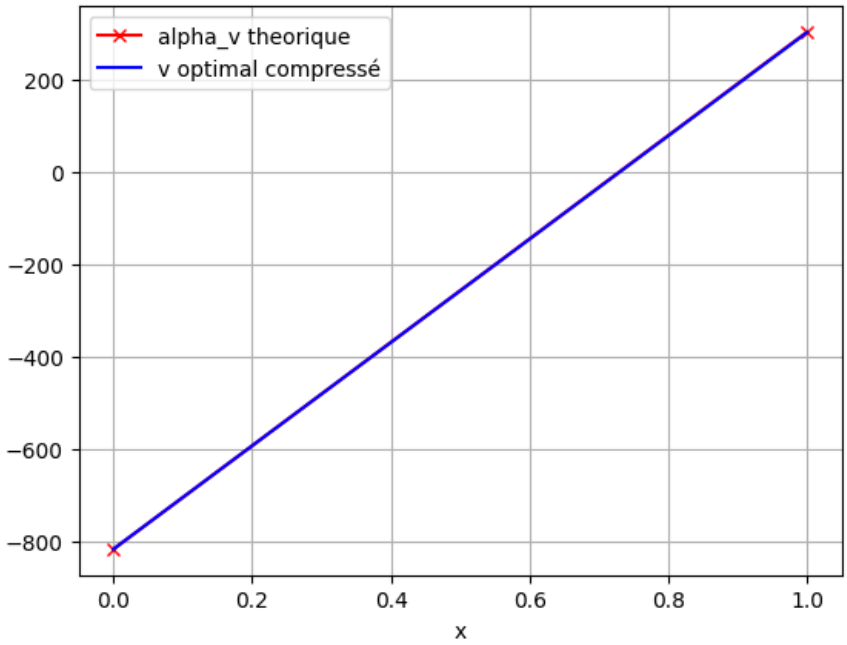

FIGURE 7 – Affichage du contrôle optimal v compressé et de la valeur théorique α_v

Étant donné que les deux résultats correspondent, on s'attend donc à ce que α_v représente bien le contrôle optimal v lors de la décompression. On s'attend à ce que α_v corresponde à $4\pi^2\, sin(2\pi\, x)$ une fois décompressé.

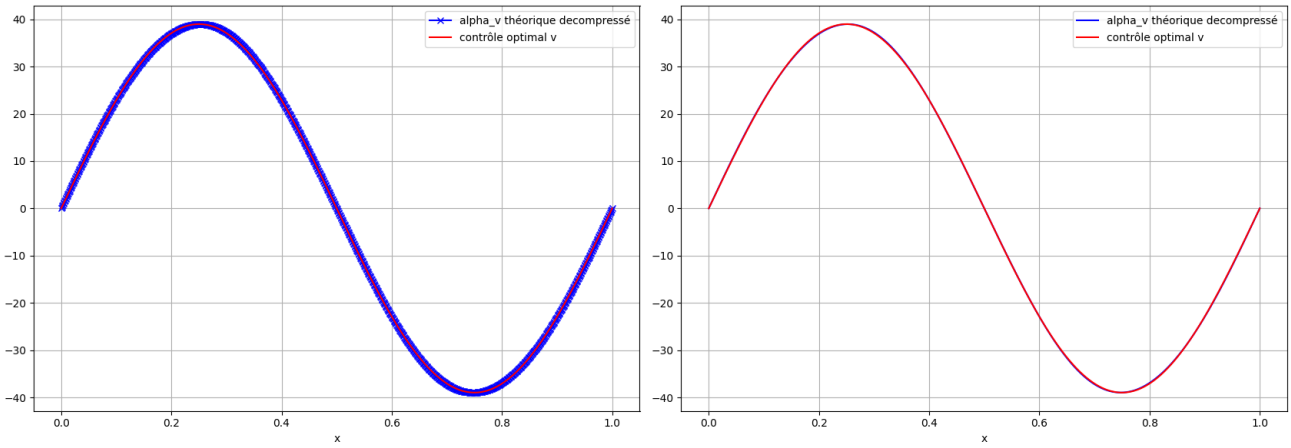

FIGURE 8 – Affichage du contrôle optimal v et de la solution α_v décompressé

La figure 8 nous montre que la solution α_v correspond bien à $4\pi^2\, sin(2\pi\, x)$ une fois décompressé c'est-à-dire que $\alpha_v \Phi_K = 4\pi^2\, sin(2\pi\, x) = v$ avec v le contrôle optimal.

Finalement, à travers l'étude de l'équation réduite nous avons réussi à retrouver le contrôle optimal. L'avantage de la réduction de dimension est l'utilisation des ressources informatiques. En effet, les calculs sont beaucoup plus légers qu'en grande dimension. Cela nous permet de gagner du temps de calcul mais aussi de limiter l'utilisation des ressources. Dans notre cas nous avons dû calculer la matrice de snapshots qui était plus coûteuse en matière de ressources que la méthode POD elle-même. Cependant, ces données sont en général fournies et de très grandes tailles car elles sont issues d'une observation physique d'où l'intérêt de travailler avec les bases réduites.

3 Apprentissage

L'apprentissage automatique est un domaine de la science qui repose sur des réseaux de neurones artificiels. C'est un champ d'études qui se fonde sur des approches mathématiques pour donner aux ordinateurs la capacité d'apprendre et de s'entraîner à partir de données qu'on lui fournit. Autrement dit, cela permet d'améliorer les performances des ordinateurs pour résoudre des solutions sans être explicitement programmés.

On appelle un réseau de neurones profond les réseaux de neurones ayant la capacité de regrouper et classifier les données stockées sur une base de données.

Les réseaux de neurones profonds sont efficaces pour fournir des approximations précises pour les fonctions en haute dimension. Cependant, il est nécessaire de posséder un grand nombre de donnée qui permettent aux réseaux de s'entraîner. Ces données ne sont pas toujours accessibles. C'est pourquoi la difficulté aujourd'hui dans l'univers de l'ingénierie est de savoir où tirer ces données manquantes. Dans cette partie nous allons proposer la méthode Neural Galerkin qui se base sur un schéma d'apprentissage profond qui génère des données d'entraînement avec un apprentissage actif pour résoudre les équations aux dérivées partielles en grande dimension. Pour cela, la méthode entraîne le réseau tout en minimisant le résidu en temps. Ceci permet de collecter de nouvelles données de manière automatique tout en étant guidé par la dynamique décrite par les équations aux dérivées partielles. Cette manière de faire est très différente des autres méthodes d'apprentissage automatique. En effet, les autres méthodes visent à ajuster les paramètres du réseau globalement dans le temps sans prendre en compte l'acquisition des données d'entraînement qui se forment au cours du temps.

Nous savons que les équations aux dérivées partielles sont utilisées pour décrire la dynamique des systèmes dans une grande variété d'applications scientifiques. Un grand nombre de ces équations ne peut être résolu en raison de leur nature complexe. Les méthodes classiques numériques que nous connaissons sont bien efficaces lorsque le domaine de la solution est de faible dimension. Cependant,

lorsqu'on passe dans des dimensions plus grandes les coûts des calculs augmentent avec un taux très élevé.

Dans ce qui suit nous allons exposer une approche numérique pour les solutions d'EDP en grande dimension. Pour cela nous développons des intégrateurs en temps pour les équations aux dérivées partielles qui utilisent des réseaux de neurones profonds pour représenter les solutions. Ensuite on met à jour des paramètres de manière séquentielle. Ces paramètres vont ensuite être réutilisés pour construire la solution approchée de notre équation aux dérivées partielles.

Faisons quelques rappels sur les réseaux de neurones.

Les réseaux de neurones sont une familles des fonctions paramétrées non-linéaires que l'on utilise pour faire de l'apprentissage. Elles nous permettent d'obtenir des résultats avec une grande précision pour les problèmes en grande dimension.

3.1 Définition d'un réseau de neurones

Un réseau de neurones est une fonction paramétrique obtenue par composition de fonction simple que l'on appelle des couches.

Définition 5 *On appelle une couche la fonction* $L : x \in \mathbb{R}^{d_i} \to y \in \mathbb{R}^{d_{i+1}}$ *définie par*

$$L_{i,i+1}(x) = \sigma(Ax + b)$$

Avec A *une matrice réelle de taille* d_i, d_{i+1} *et* $\sigma()$ *une fonction non-linéaire qu'on appelle la fonction d'activation. Les coefficients* A *et* b *sont appelés paramètres entraînables.*

Définition 6 *On appelle un réseau de neurones la fonction paramétrique* N_θ : $x \in \mathbb{R}^d_{in} \to y \in \mathbb{R}^{d_0}$ *définie par :*

$$N_\theta(x) = L_{0,n} \circ ... \circ L_{i+1,i} \circ ... \circ L_{1,in}(x)$$

et θ *est l'ensemble des paramètres entraînables.*

On dit qu'un réseau de neurones est totalement connecté s'il s'agit d'un réseau où les matrices A sont pleines. On dit qu'une couche est cachée si elle ne concerne pas l'espace de sortie. L'ingrédient principal des réseaux de neurones est la fonction d'activation. Il en existe différents types et chacune est propre au problème que l'on a, c'est-à-dire à la condition initiale, de bords ou le domaine de la solution.

3.2 Neural Galerkin

On se donne un domaine $\Omega \subset \mathbb{R}^d$ et $\mathcal{U}$ un ensemble de fonctions où u y appartient. On veut observer l'évolution du champ $u : [0, \infty[\times\Omega \to \mathbb{R}$ où la dynamique est représentée par :

$$\begin{cases} \partial_t u(t,x) = f(t,x,u) \quad (t,x) \in [0,\infty[\times\Omega \\ u(0,x) = u_0(x) \quad x \in \Omega \end{cases} \tag{12}$$

Le terme $f : [0, \infty[\times\Omega \times \mathcal{U}$ peut représenter plusieurs équations comme par exemple l'équation de la chaleur, de diffusion ou de transport. On supposera dans ce qui suit qu'on aura les conditions de bords appropriées de sorte que l'équation (12) soit bien définie pour tout $t \in [0, \infty[$, que la solution existe qu'elle est unique et qu'elle dépend continûment de u_0.

L'objectif est de paramétrer la solution $u(t)$ comme étant $U(\theta(t)) \in \mathcal{U}$ avec les paramètres $\theta \in \Theta$ et $U : \Theta \times \Omega \to \mathbb{R}$.

Pour cela on utilise l'approche suivante $u(t,x) = U(\theta(t), x) \quad (t,x) \in [0,\infty[\times\Omega$ avec $\theta(t)$ qui est le vecteur des paramètres ajustables dans le réseau.

Il est possible que U dépende en $\theta(t)$ de manière non-linéaire ce qui fait la particularité de cette méthode. Cependant, pour avoir une bonne représentation de $u(t,x) = U(\theta(t), x)$, $(t,x) \in [0,\infty[\times\Omega$, c'est-à-dire que pour tout $u(t)$ il existe au moins un $\theta(t) \in \Theta$ tel que l'égalité est vérifiée. On devrait alors avoir Θ qui est lui même un espace fonctionnel, donc qu'il soit potentiellement de dimension infinie. Dans notre cas on va s'intéresser aux situations où la représentation paramétrique U dépend d'un paramètre de dimension finie.

Puisque nous n'avons pas accès à la solution nous avons besoin de contrôler le résidu en temps. En supposant que $\theta(t)$ est différentiable on peut injecter la solution $U(\theta(t))$ dans l'équation (12). On a alors $\partial_t U(\theta(t)) = \nabla_\theta U(\theta) \cdot \dot{\theta}(t)$. Ceci nous donne le résidu de la manière suivante :

$$r_t(\theta, \eta, x) = \nabla_\theta U(\theta, x)\eta - f(t, x, U(\theta)) \tag{13}$$

Avec $r : \Theta \times \dot{\Theta} \times \Omega \to \mathbb{R}$ avec $\dot{\Theta}$ l'ensemble des dérivées en temps de $\theta(t)$. Le moyen classique pour contrôler le résidu est de le minimiser selon $\eta \in \dot{\Theta}$. On le fait de manière locale en temps c'est-à-dire on cherche $\theta(t)$ pour $t > 0$ de la manière suivante :

$$\dot{\theta(t)} \in \underset{\eta \in \dot{\Theta}}{argmin} J_t(\theta(t), \eta) \tag{14}$$

avec la fonctionnelle coût $J_t(\theta(t), \eta) : \Theta \times \dot{\Theta} \to \mathbb{R}$ définie comme suit :

$$J_t(\theta, \eta) = \frac{1}{2} \int_\Omega |r_t(\theta, \eta, x)|^2 d\nu(x) \tag{15}$$

avec ν une mesure positive à support dans Ω.

Puisque $J_t(\theta, \eta)$ est différentiable et convexe, le minimum de J_t résout l'équation de Euler-Lagrange :

$$\nabla_\eta J_t(\theta(t), \dot{\theta(t)}) = 0 \tag{16}$$

Il peut être difficile de résoudre $U(\theta(t), x)$ c'est pourquoi on pose le système d'équation différentielle ordinaire

$$\begin{cases} M(\theta)\dot{\theta} = F(t, \theta), \theta(0) = \theta_0 \\ M(\theta) = \displaystyle\int_\Omega \nabla_\theta U(\theta, x) \otimes \nabla_\theta U(\theta, x) d\nu \\ F(t, \theta) = \displaystyle\int_\Omega \nabla_\theta U(\theta, x) f(t, x, U(\theta)) d\nu \end{cases} \tag{17}$$

Où $\otimes$ désigne le produit extérieur et ν une mesure ayant son support dans Ω. L'équation (17) sera la base des schémas numériques qu'on introduira plus tard. $U(\theta(t))$ sera nommée la solution de Neural Galerkin étant donné que (17) peut être dérivée en utilisant $\nabla_\theta U(\theta(t))$ comme fonction test et en utilisant la projection de Galerkin par rapport au produit scalaire dans $L^2(\Omega, \nu)$.

3.3 Estimation de $M(\theta)$ et $F(t, \theta)$

Pour un choix générique de paramètres $U(\theta)$ les intégrales de l'équation (17) ne peuvent pas être calculées de manière traditionnelle. En effet, lorsque Ω possède une dimension élevée il faut approcher les valeurs des intégrales par des méthodes numériques comme celle de Monte-Carlo. Il s'agit de se donner un échantillon $\{x_i\}_{i=1}^n$ distribué aléatoirement selon une loi de probabilité ν et de calculer les moyennes empiriques sur ces échantillons.

On se donne une mesure de probabilité ν informée par la solution $U(\theta)$. On utilise la solution $U(\theta)$ pour construire la manière dont les échantillons $\{x_i\}_{i=1}^n$ sont attribuées. On a alors par la méthode de Monte-carlo :

$$\begin{array}{l} \widetilde{M}(\theta) = \frac{1}{n}\sum_{i=1}^n \nabla_\theta U(\theta, x_i) \otimes \nabla_\theta U(\theta, x_i) \\ \widetilde{F}(t,\theta) = \frac{1}{n}\sum_{i=1}^n \nabla_\theta U(\theta, x_i) f(t, x_i, U(\theta)) \end{array} \tag{18}$$

3.4 Discrétisation en temps

Maintenant que nous avons un moyen de calculer les intégrales de manière numérique, nous pouvons approcher $M(\theta)\dot{\theta} = F(t,\theta)$. On note par θ^k l'approximation numérique de $\theta(t_k)$ avec t_k qui sont définis de la manière suivante : $t_0 = 0$, $t_{k+1} = t_k + \delta_{t_k}$ avec $\delta_{t_k} > 0$ un pas de temps.

Nous voulons à chaque pas de temps mettre à jour nos paramètres θ^k. Pour cela nous pouvons le faire manière explicite ou de manière implicite. La manière explicite s'écrit alors :

$$\widetilde{M}(\theta^k)\theta^{k+1} = \widetilde{M}(\theta^k)\theta^k - \delta_{t_k}\widetilde{F}(t_k, \theta^k) \tag{19}$$

Et la manière implicite est :

$$\widetilde{M}(\theta^{k+1})\theta^{k+1} = \widetilde{M}(\theta^{k+1})\theta^k - \delta_{t_k}\widetilde{F}(t_{k+1}, \theta^{k+1}) \tag{20}$$

3.5 Architecture de réseaux de neurones

Maintenant qu'on possède les outils pour mettre à jour les θ^k et calculer les valeurs des intégrales de (17) on peut utiliser l'apprentissage profond pour paramétrer $U(\theta, x)$.

3.5.1 Calcul de $U(\theta)$

Pour résoudre l'équation (17) nous utilisons une architecture neurale avec une couche cachée et m mode. En d'autres termes

$$U(\theta, x) = \sum_{i=1}^{m} c_i \varphi(x, w_i, b_i) \tag{21}$$

où θ est définie comme $\theta = (c_i, w_i, b_i)_{i=1}^{m}$ avec $c_i, w_i, b_i \in \mathbb{R}$ sont les paramètres du réseau de neurones profond et $\varphi : \Omega \times \mathbb{R} \times \mathbb{R}$ est une fonction d'activation. Le m dépend du nombre de fonctions d'activation que l'on choisit.

3.5.2 Exemple numérique de Neural Galerkin

Pour finir cette partie nous allons implémenter cette méthode. On se donne l'équation suivante :

$$\begin{cases} \partial_t u(t,x) = D\partial_{xx} u(t,x) - a\partial_x u(t,x) \quad dans\,[0,1] \\ u(t_0) = \dfrac{1}{\sqrt{2\pi t_0}} exp(-\dfrac{1}{2t_0}(x - at_0 - x_0)^2) \\ \partial_x u(0,t) = \partial_x u(1,t) = 0 \end{cases} \tag{22}$$

Avec D le coefficient de diffusion et a l'advection. Dans notre cas nous allons choisir $D = 0$ et $a = 1$. On obtient donc une équation de transport.

Il est possible de résoudre cette équation en utilisant les différences finies en temps. Nous allons utiliser le résultat obtenu pour le comparer avec le résultat donné par la méthode de Neural Galerkin. On obtient la solution qui est représentée dans la figure suivante :

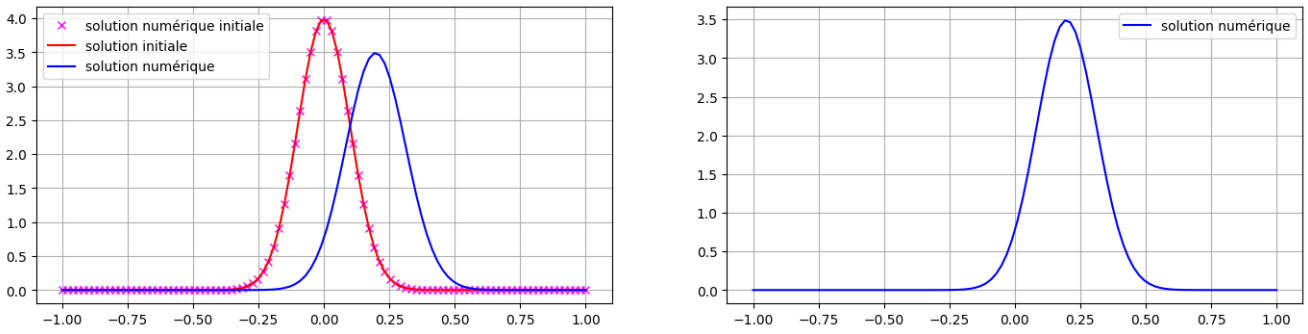

FIGURE 9 – Affichage de la solution de l'équation (22) par différences finies

La solution donnée par Neural Galerkin devra donc approcher la solution affichée en bleu dans la figure 9.

On choisit comme fonction d'activation la Gaussienne suivante :

$$G(x, b_i, w_i) = c_i exp(-\frac{1}{2}\frac{(x-b_i)^2}{w_i})$$

On pose alors le paramètre $\theta = (c_i, b_i, w_i)_{i=1}^m$. On décide de mettre à jour les θ^k selon le schéma explicite. Par conséquent, la solution de l'équation (22) sera donnée par

$$U(\theta, x) = \sum_{i=1}^{m} c_i G(x, b_i, w_i)$$

Il nous faut calculer le gradient par rapport à θ pour être capable de mettre à jour les θ décrit par la relation (19). Nous avons alors :

$$\nabla_\theta U(\theta, x) = \begin{bmatrix} \nabla_{c_i} U(\theta, x) \\ \nabla_{\omega_i} U(\theta, x) \\ \nabla_{b_i} U(\theta, x) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m G(x, b_i, \omega_i) \\ \sum_{i=1}^m \frac{c_i (x-b_i)^2}{\omega_i^2} G(x, b_i, \omega_i) \\ \sum_{i=1}^m \frac{c_i (x-b_i)}{\omega_i} G(x, b_i, \omega_i) \end{bmatrix}$$

On choisit le pas de temps $\delta_{t_k} = 0.185$. L'échantillon $\{x_i\}_{i=1}^n$ est distribué selon une loi normale de moyenne nulle et d'écart-type 1. Observons les résultats obtenues pour différentes valeurs de n.

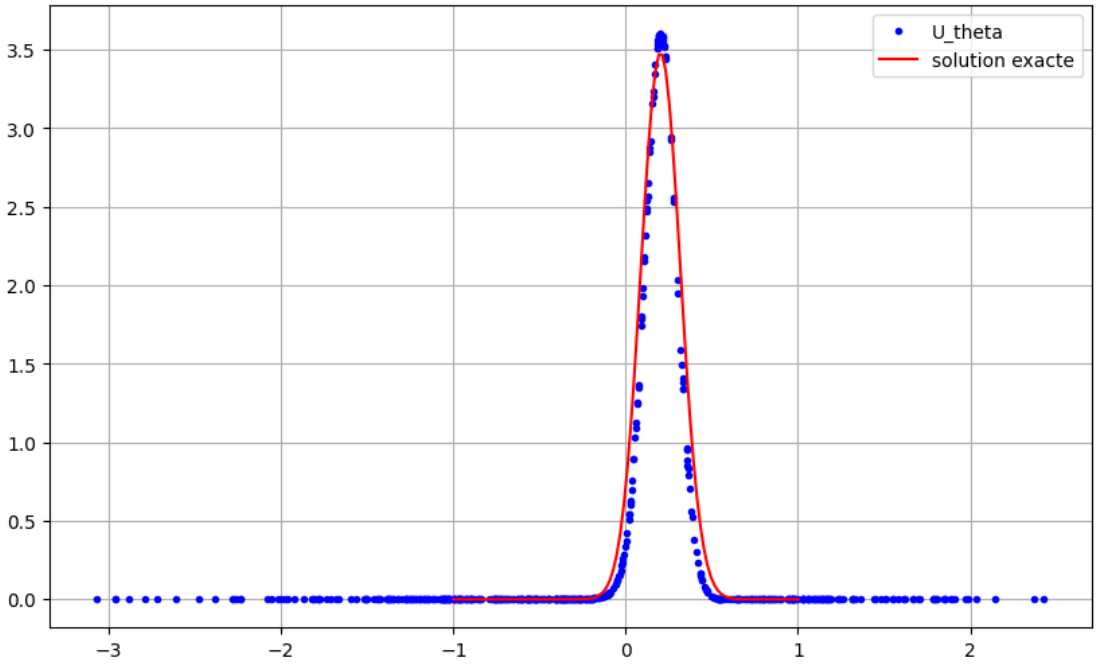

FIGURE 10 – Affichage de la solution calculée par la méthode de Neural Galerkin avec $n = 500$

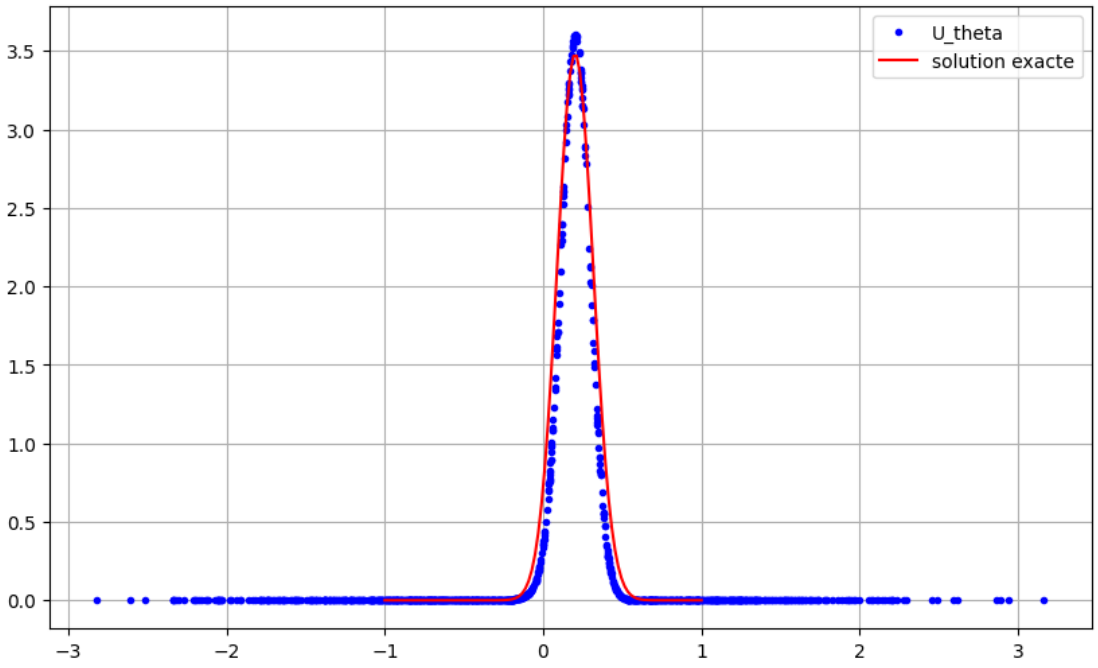

FIGURE 11 – Affichage de la solution calculée par la méthode de Neural Galerkin avec $n = 1000$

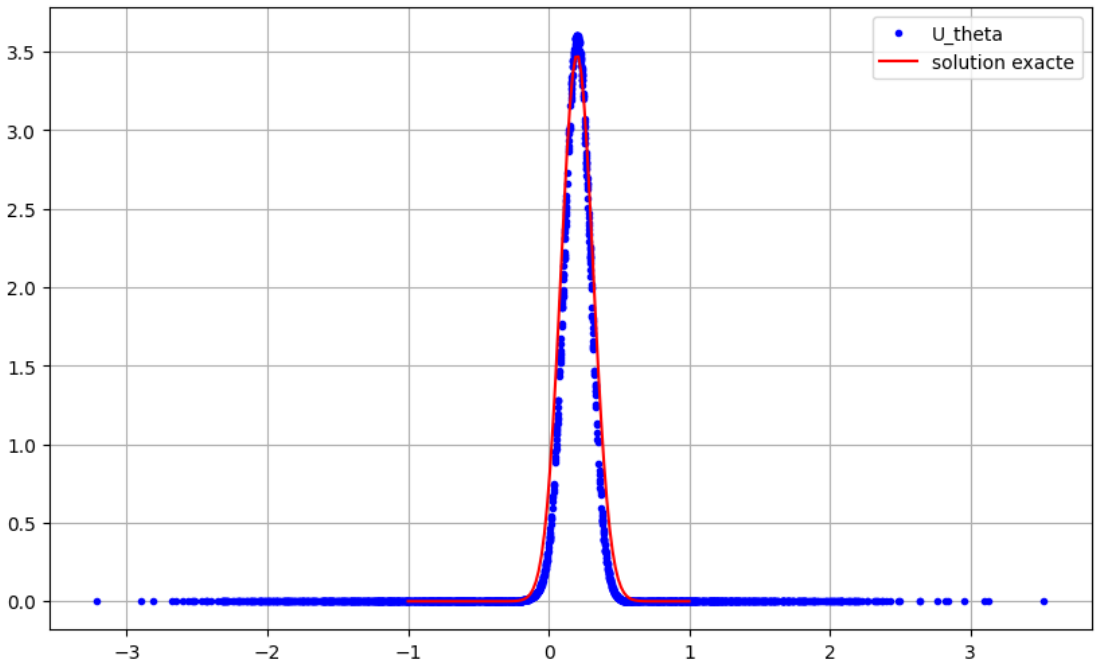

FIGURE 12 – Affichage de la solution calculée par la méthode de Neural Galerkin avec $n = 2000$

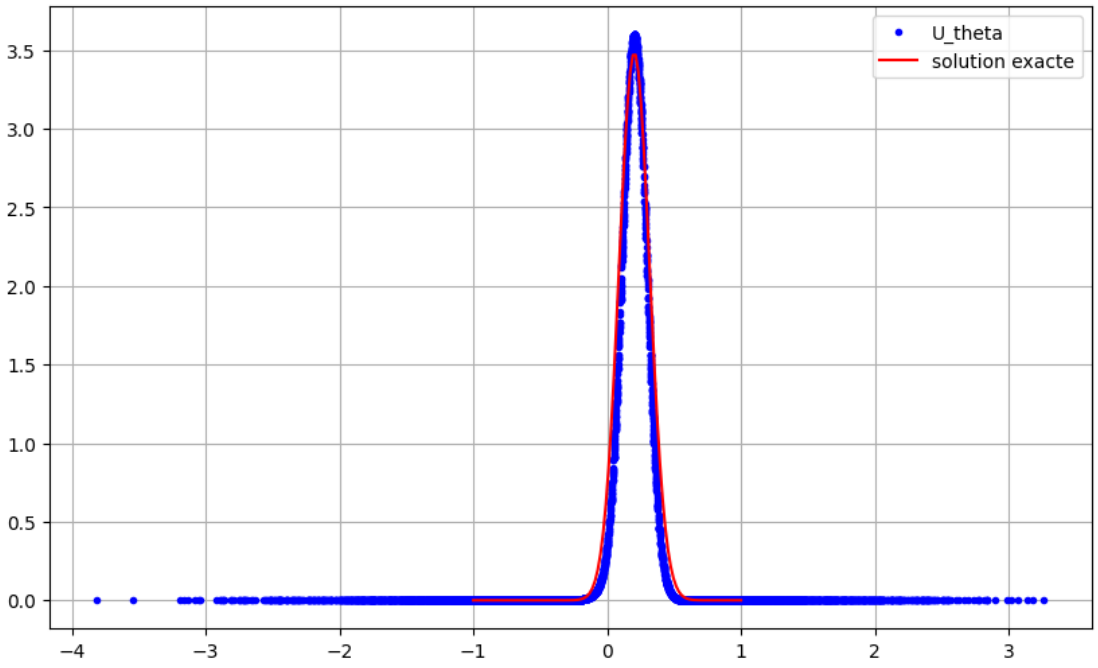

FIGURE 13 – Affichage de la solution calculée par la méthode de Neural Galerkin avec $n = 5000$

Puisque nous avons choisi une seule Gaussienne comme fonction d'activation on pose alors $m = 1$ dans la relation (21). On remarque que la méthode Neural Galerkin a tendance à approcher la solution lorsqu'on distribue plus de points $\{x_i\}_{i=1}^{n}$. Ceci nous montre que malgré le fait d'utiliser une seule Gaussienne il est possible d'approcher correctement la solution exacte qui nous a été fournie par la méthode des différences finies.

Conclusion

Dans cette étude nous avons analysé trois manières d'approcher une solution d'une équation aux dérivées partielles. Chacune des méthodes est propre au problème qu'on se donne ou à l'équation qu'on considère. Dans le cadre du contrôle optimal nous avons montré qu'il existe un moyen d'exprimer des conditions d'optimalités d'une fonction coût qui est associée à une équation aux dérivées partielles. Il s'agit de la méthode d'adjoint. Le fait d'introduire l'équation adjointe nous permet de calculer le gradient de la fonction coût et donc d'obtenir des conditions d'optimalités qui nous sont utiles pour déterminer le contrôle optimal. En effet, nous avons utilisé une méthode de gradient à pas constant pour déterminer le contrôle optimal de l'équation (1). Par conséquent, le gradient nous a été utile pour mettre à jour la descente de la méthode. Ceci nous a permis d'obtenir une approximation numérique du contrôle optimal.

Nous avons montré qu'il était possible d'approcher la solution du contrôle optimal obtenue par la méthode du gradient en travaillant dans un espace réduit. La particularité de la réduction de dimension pour les équations aux dérivées partielles est la capacité de renvoyer une solution approchée de manière précise avec un nombre de données beaucoup plus petit qu'en grande dimension. Pour cela, il était essentiel de bien projeter nos données dans notre espace réduit afin de faire la minimisation de la fonction coût dans cet espace. Nous avons décidé de prendre une matrice de snapshots de taille 1000x1000 et de travailler avec une méthode POD pour la réduction. Lorsque nous avons calculé l'opérateur Φ_K qui permet de compresser et de décompresser nos données, nous avons pu extraire l'information du plus petit nombre de modes nécessaires pour réduire notre modèle, comme c'est montré dans la figure 6. Nous avons ensuite fait la compression de notre modèle et nous avons calculé le contrôle optimal. Une fois le contrôle optimal réduit décompressé, on trouvait le même résultat que pour la méthode de gradient. Le fait d'avoir travaillé dans un espace réduit nous a permis de gagner du temps de calcul mais aussi de conserver les ressources numériques. En effet, dans la méthode du gradient on résout beaucoup d'équations duales et primales ce qui augmente le temps de calcul, notamment lorsque la dimension est élevée.

La dernière méthode pour résoudre des équations aux dérivées partielles se base sur l'apprentissage profond. Nous avons discuté de manière théorique du schéma Neural Galerkin. L'intérêt de cette méthode est de faire l'acquisition des données et des simulations de manière adaptative. Le système apprend la solution de l'EDP en faisant évoluer une représentation d'une fonction non-linéaire qui est ajustée à partir des points de donnée issus d'un échantillon d'une mesure de probabilité qui est informée par la solution. Cette méthode devient alors intéressante lorsque nous travaillons avec un problème en grande dimension. En effet, on peut écrire l'équation (17) à la place de notre équation aux dérivées partielles. De là on peut alors utiliser des méthodes d'intégration numérique basiques telles que la méthode de Monte-Carlo pour approcher les intégrales $M(\theta) = \int_\Omega \nabla_\theta U(\theta, x) \otimes \nabla_\theta U(\theta, x) d\nu$ et $F(t, \theta) = \int_\Omega \nabla_\theta U(\theta, x) f(t, x, U(\theta)) d\nu$. Ces approximations nous sont utiles car

on décide de représenter la solution $u(t,x) = U(\theta(t),x)$ avec $\theta(t)$ le paramètre qui va évoluer en temps dans notre méthode. On décide alors de le mettre à jour par la relation (19) et d'exprimer la solution de la manière suivante : $U(\theta,x) = \sum_{i=1}^{m} c_i G(x,b_i,w_i)$, avec $\theta = (c_i,b_i,w_i)_{i=1}^{m}$. On a vérifié cette méthode en résolvant l'équation (22). D'abord nous avons effectué une méthode de différences finies en temps pour comparer avec la méthode Neural Galerkin. Nous avons pris une seule Gaussienne comme fonction d'activation et donc $m = 1$. On a décidé de distribuer les points $\{x_i\}_{i=1}^{n}$ selon une loi normale de moyenne 0 et d'écart type 1. On remarque alors que la méthode approche correctement notre solution obtenue par les différences finies.

Annexes

Toutes les implémentions sont faites en python

```
#différences finies
def DF(n,f):

    A = 2*np.eye(n)-np.diag(np.ones(n-1),1)-np.diag(np.ones(n-1),-1) + h**2*np.eye(n) #matrice A

    A[0][0] = 1
    A[0][1] = 0
    A[n-1][n-1] = 1
    A[n-1][n-2] = 0

    F = h**2*f

    y = solve(A,F) #on résout le systeme Ay=F
    return y

#méthode de gradient à pas constant
def Gradient(x,J,grad_J, eps, rhs_f,zd, v0, kmax, alpha, pas):

    res = 1
    k = 0

    Res = np.zeros(kmax) #matrice pour observer l'évolution du résidu
    J_plot = np.zeros(kmax) #matrice qui nous permet d'observer les valeurs de J à chaque étape
    V_plot = np.zeros((kmax,kmax)) #matrice pour observer l'évolution du contrôle

    while(res> eps and k<kmax):

        V_plot[k] = v0
        y = DF(N,rhs_f+v0)
        J_plot[k] = J(y,zd,alpha,v0)
        p = DF(N,y-zd)
        dk = -grad_J(p,alpha,v0)#direction
        v = v0 + pas * dk
        res = norm(v-v0)
        Res[k] = res
        v0 = v
        k+=1
    return y, p, v0, Res, -dk, J_plot, V_plot
```

```
def psi(mesh):
    psi1 = np.sin(2*np.pi*mesh)
    psi2 = np.sin(2*2*np.pi*mesh)
    psi3 = np.sin(2*3*np.pi*mesh)
    psi4 = np.sin(2*4*np.pi*mesh)
    psi5 = np.sin(2*5*np.pi*mesh)

    return psi1,psi2,psi3,psi4,psi5

#matrice de snapshots
def base_de_donnee(N,mesh):

    psi1,psi2,psi3,psi4,psi5 = psi(mesh)
    Psi = [psi1,psi2,psi3,psi4,psi5]

    u = np.zeros((N,500))
    s = np.zeros((N,500))

    for i in range (500):
        res = np.zeros(N)
        ei = np.random.uniform(-1,1,5)
        for i in range (4):
            res = res + ei[i]*Psi[i]
        s[:,i] = res[:]
        u[:,i] = DF(N,s[:,i])

    Ms = np.column_stack((s,u))

    return Ms

#Méthode POD
def reduction_POD(snapshots,K_modes):
    snap = snapshots
    snap0 = cp.copy(snapshots)
    ref = np.zeros_like(snapshots[:,0])

    for i in range(0,snapshots.shape[1]):
        snap[:,i]=snap0[:,i]-ref[:]


    #on fait la décomposition SVD
    U, s, _ = sp.linalg.svd(snap, full_matrices=False)

    # Extraire les premiers K vecteurs singuliers de U
    phi_K  = U[:, :K_modes]

    return phi_K, s
```

```
def Solve_Differences_Finis(v, nt, nx, s, D, a, tf, dt,x,x0):
    dx = 1 / (nx - 1)
    Us = np.zeros((nt + 1, nx))
    Us[0, :] = v.copy()
    Time = []
    time = 0
    k = 0
    t=1
    exacte = []
    while(time < tf):
        un = v.copy()

        for i in range(1, nx - 1):
            v[i] = un[i] + D*s*(un[i+1]-2*un[i]+un[i-1])-a*s*dx*(un[i]-un[i-1])

        Us[t, :] = v.copy()
        t = t+1
        exacte.append(gaussienne(a,time,t0,x,x0))
        time += dt
        Time.append(time)
        k = k + 1

    return Us, k, Time, np.array(exacte)




def dU_x(x,t,a,b,w,x0):
    e = np.exp(-(x-a*t-x0)**2/(2*(t)))
    return - (1/np.sqrt(2*np.pi*(t))) * ((x-a*(t)-x0)/(t)) * e




def dU_xx(x,t,a,b,w,x0):

    e = np.exp(-(x-a*t-x0)**2/(2*(t)))
    return -(1/np.sqrt(2*np.pi*(t))) * (1/(t)) * e + 1/np.sqrt(2*np.pi*(t)) * ((x-a*t-b)/(t))**2 * e
```

```
def Gauss(x,c,b,w):
    return c*np.exp(-(1/(2*w))*((x-b)*(x-b)))




def Grad_U_theta(x,c,b,w):
    grad_ci = Gauss(x,1,b,w)
    grad_wi = (x-b)*(x-b)/(w*w) * Gauss(x,c,b,w)
    grad_bi = (x-b)/w * Gauss(x,c,b,w)

    Grad_theta = [grad_ci,grad_bi,grad_wi]

    return np.array(Grad_theta)




#calcul de M(theta) = 1/n * somme_i^n Grad_Theta U(theta) x Grad_theta U(theta)
#calcul de F(Theta) = 1/n * somme_i^n Grad_Theta * f(t,xi,U(theta))
def Mc(n,x,c,b,w,D,a,t,x0):

    M_theta = 0
    F_theta = 0
    f = 0
    for i in range(n):

        xi = x[i]
        Grad = Grad_U_theta(xi,c,b,w)
        ProduitExt = np.outer(Grad,Grad)
        f = -D*dU_xx(xi,t,a,b,w,x0) + a*dU_x(xi,t,a,b,w,x0)
        M_theta = M_theta + ProduitExt
        F_theta = F_theta + np.dot(Grad, f)

    return np.dot((1/n),M_theta), np.dot((1/n),F_theta)
```

```
def theta(D,a,t,c,b,w,it,n,delta_t,x,x0):

    Theta = []
    theta = np.array([c,b,w]) #theta_0
    Theta.append(theta)

    t_k = t

    M_theta0,F_theta0 = Mc(n,x,c,b,w,D,a,t_k,x0) #calcul de M(theta_0)


    for _ in range (it):

        #calcul de M(theta_k) theta_{k+1} = M(theta_k)*theta_k - delta_tk*F(t_k,theta_k)
        #theta_k =  (theta - delta_t*np.linalg.inv(M_theta0)@F_theta0)
        delta_theta = np.linalg.solve(M_theta0, np.dot(delta_t, F_theta0))

        theta_k =theta - delta_theta

        t_k += delta_t #t_{k+1} = t_k + delta_tk

        c = theta_k[0]
        b = theta_k[1]
        w = theta_k[2]

        M_theta0, F_theta0 = Mc(n,x,c,b,w,D,a,t_k,x0) #M(theta_k)

        theta = theta_k
        Theta.append(theta)
    return np.array([Theta])

#calcul de U(theta,x) = somme_{i=1}^n c_i G(x_i,b_i,w_i)
def U_theta(theta,m,x):

    U_thet = 0

    for i in range (m):
        c = theta[0,i][0]
        b = theta[0,i][1]
        w = theta[0,i][2]
        U_thet+=Gauss(x,c,b,w)


    return U_thet
```

Références

-[1] Haïm Brezis (2020), *Analyse fonctionelle, théorie et applications*, Dunod,11 rue Paul Bert, 92240 Malakoff

-[2] Joan Bruna, Benjamin Peherstorfer, Eric Vanden-Eijnden, (2022) *«Neural Galerkin Scheme with Active Learning for High-Dimensional Evolution Equations»*, arXiv :2203.01360v3.

-[3] Yannick Privat (2023), *«Problèmes de contrôle optimal impliquant les EDP»*, IRMA Université de Strasbourg, `https://irma.math.unistra.fr/~privat/documents/M2_plan/optimM2_CO_plan_chap7.pdf`

-[4] Yannick Privat (2023), *Rappels sur la régularité des ouverts et les équations elliptiques*, IRMA Université de Strasbourg, `https://irma.math.unistra.fr/~privat/documents/M2_plan/regularite_bound.pdf`

-[5] Yannick Privat, *Méthodes d'adjoint pour le contrôle optimal sous contrainte EDP*, IRMA Université de Strasbourg, `https://irma.math.unistra.fr/~privat/documents/M2_CO/coursEDP.pdf`

-[6] Christophe Prud'homme,(2023), *Reduced Basis approximation*, IRMA Université de Strasbourg

-[7]Christophe Prud'homme,(2023), *A posteriori error estimation*, IRMA Université de Strasbourg

-[8] Emmanuel Franck,(2023), *Réduction de dimension pour les EDP*, IRMA Université de Strasbourg, `https://irma.math.unistra.fr/~franck/cours/SciML/output/html/chapRED_sec1.html`

-[9] Emmanuel Franck,(2023), *«Introduction aux réseaux de neurones»*, IRMA Université de Strasbourg, `https://irma.math.unistra.fr/~franck/cours/SciML/output/html/chapAP_sec1.html`

-[10] Sébastien Tordeux et Victor Péron, (2020-2021), *«Analyse numérique, la méthode des différences finies»*, Université de Pau, `https://stordeux.perso.univ-pau.fr/COURS/AN1.pdf`

-[11] Charbel Farhat, *« MODEL REDUCTION. Proper Orthogonal Decomposition (POD)»*, Stanford University, `https://web.stanford.edu/group/frg/course_work/CME345/CA-CME345-Ch4.pdf`

-[12] Christophe Prud'homme,(2023), *«SVD and POD»*, IRMA Université de Strasbourg

-[13] B. Helffer, (2007), *«Introduction aux Equations aux Dérivées Partielles»*, `https://www.imo.universite-paris-saclay.fr/~bernard.helffer/coursS4edp07.pdf`